Mai 1988
7 DM · 60 öS · 7 sFr

Die Mikrocomputer-Zeitschrift

5/88

Schwerpunkt:

Computer-Grafik

Bilder aufbereiten
EGA-Palette
transparent
CGA-Emulation
auf Hercules

Test:
Overlays unter
Turbo-Pascal 4.0
MASM 5.0
Textmaker 1.0

Serien:
Superprozessor 68030
Einführung in Intuition
Wie man den 80287 nutzt
Mit Modula programmieren

C-Funktionen
im Amiga-ROM

Hardware:
Omega-Net, ein
PC-Peripherie-Netz

M. L. & S.

EPSON. Der Unterschied.

Wie schön, daß bei uns große Leistungen immer so handlich ausfallen. Der neue EPSON Laser-Drucker GQ-3500.

Unser neuer Laser-Drucker zeichnet sich schon auf den ersten Blick durch seine handlichen Abmessungen aus. Moderne Laser-Drucktechnik ist damit an jedem Arbeitsplatz einsetzbar. Mit all ihren Vorzügen: Niedriges Arbeitsgeräusch, hohes Arbeitstempo, gestochen scharfe Druckwiedergabe.

Doch die entscheidenden Unterschiede stecken innen: Das EPSON Steuersystem „ESC/P Page", das für größtmöglichen Komfort bei der Aufbereitung der Druckseiten sorgt.

Mit feststehenden Macro-Befehlen für vielfältige Gestaltungselemente. Und das IC-Steckkarten-System, mit dem sich die reichhaltige Schriften-Auswahl im Handumdrehen erweitern läßt. Es vereinfacht aber auch die Drucker-Anpassung. So können Sie auch all die Anwendungs-Programme nutzen, in denen andere weitverbreitete Drucker berücksichtigt sind, wie HP Laserjet PLUS oder Diablo 630. Diese großen Leistungen erhalten Sie zu einem Preis, der das Zugreifen leicht macht.

Schriftenvielfalt im Scheckkarten-Format.

EPSON

Technologie, die Zeichen setzt.

EPSON Deutschland GmbH · Zülpicher Straße 6 · 4000 Düsseldorf 11 · Telefon 0211/5603-0
Vertriebsbüro Hamburg: Telefon 040/441331-34 · Vertriebsbüro München: Telefon 089/917205-07

EDITORIAL

Liebe Leser!

mc wird genau gelesen. Ich hatte in der Modula-Serie eine kleine Bemerkung gemacht, daß Niklaus Wirth eine neue Sprache namens Oberon entworfen hätte. Nur diese Mitteilung allein genügte vielen, um zu fragen, wo man mehr Informationen über Oberon bekommen könnte. Die Sachlage ist kompliziert. Es gibt ein hochschulinternes Papier in Zürich, „From Modula to Oberon" zusammen mit „The Programming Language Oberon", in dem Oberon beschrieben ist. Das Papier ist leider vergriffen. Wer an die Informationen kommen muß, kann versuchen, in der nächstgelegenen Universitätsbibliothek das Script auszuleihen. Es muß allerdings eine Hochschule mit starker Informatik sein. Denn in Zürich wurde mir gesagt, daß das Papier von vielen solchen Hochschulen angefordert worden sei. Allerdings wird das Papier baldigst in der Zeitschrift „Software Praxis and Experience" (J. Whiley) erscheinen. Das wird das eigentliche Dokument sein, denn dort wird der neueste Stand veröffentlicht werden. Ein Zitat aus dem Abstract von N. Wirth: „Die Programmiersprache Oberon ist das Ergebnis einer konzentrierten Anstrengung, die Leistungsfähigkeit von Modula-2 zu steigern und gleichzeitig die Komplexität zu verringern. Einige Elemente wurden weggelassen und einige wenige hinzugefügt, um die Ausdrucksfähigkeit und Flexibilität der Sprache zu erhöhen." Oberon ist allerdings erst im Entwicklungsstadium, und das wird noch einige Jahre dauern. Jetzt ist zunächst Modula so weit, daß man darauf vertrauen kann, daß der große Pool an Modula-Programmen ständig wächst und eine Eigendynamik entwickeln wird, wie es bei Pascal war und auch heute noch ist.

Schwerpunkt in diesem Heft ist Computergrafik. Dabei handelt es sich hier nicht nur um einfache Liniengrafik. In einem schönen Aufsatz beschreiben die Autoren H.-W. Eisermann und K. D. Brinkmann, wie man mit (viel) Mathematik Bilder im Computer verbessern oder verfremden kann. Es ist erstaunlich, welche Effekte erzielbar sind. Mit Fourier-Transformation und Phantasie ins Reich der Bilder! Passend dazu mag der Beitrag „Arbeiten mit dem 80287" sein, der in diesem Heft ganz konkret aufzeigt, wie man den Coprozessor vom Assembler her oder von C aus anspricht. Es ist Tatsache, daß die tiefergehende Erarbeitung oder Bearbeitung von Bildmaterial im Computer nur unter großem Rechenaufwand gelingt. Da ist der Einsatz eines Floating-Point-Prozessors zwingend notwendig.

Ray Duncan hat uns den zweiten Brief aus den USA geschickt. Für alle, die ihn nicht kennen: Ray Duncan ist einer der Top-MS-DOS- und Mikrocomputerspezialisten in den USA. Zum Beispiel hat er das Werk „The MS-DOS Encyclopedia" als „General Editor", also als Chef-Lektor betreut. Dieses Werk ist gerade jetzt bei Microsoft Press (in Englisch) erschienen und dürfte das vollständigste sein (1570 Seiten!), das auf dem Markt ist. Es wendet sich an Programmierer, die wirklich in die Vollen gehen müssen oder wollen. Für uns hat Ray Duncan diesmal die Stellung des 80386 im Wertsystem der Power User und Power-Programmmierer in den USA kommentiert. Mit pikanten Details aus der Entwicklungsgeschichte von OS/2!

Für Leute mit ganz praktischen Steuerungsproblemen, denen das Betrachten noch so leistungsfähiger Prozessoren oder Programmiersprachen nichts hilft, gibt es in diesem Heft einen Hardware-Beitrag. Wir stellen ein Peripherie-Netz vor, mit dem preiswerte Steuerungssysteme aufgebaut werden können. Universell für kleine Anwendungen: Omega-Net. Das Netzwerk nutzt modernste Tokenring-Technik. Viel Freude an mc!

Ihr

Ulrich Rohde

DIE UNIC-ICB-IDEE

XT/AT-KOMPATIBLE COMPUTER IN INDUSTRIELLEM DESIGN

Die starke Verbreitung von IBM-kompatiblen Rechnersystemen hat es mit sich gebracht, daß eine sehr große Anzahl von entsprechenden Betriebssystemen, Programmiersprachen, Utilities und Programmgeneratoren rasch und preisgünstig erhältlich ist. Die Verwendung dieser Programme für den industriellen Einsatz (Maschinensteuerung, Betriebsdatenerfassung) scheiterte meist daran, daß normale Rechnersysteme für rauhe Umgebungsbedingungen nicht ausgelegt waren (erschütterungsempfindlich, Probleme mit Staub, Temperatur und Feuchtigkeit).

VORTEILE DER UNIC-ICB-RECHNER

100 % softwarekompatibel zu IBM XT/AT-Rechnern. Vollständiger Aufbau auf Doppel- und Einfach-Europakarten – genormte Gehäuse, beliebige Schutzarten. Erweiterter Temperaturbereich durch entsprechende Steckverbindungen und Bauteileauswahl. Erschütterungsunempfindlich durch Einsatz nichtrotierender Massenspeicher (RAM/EPROM-Floppy).
Vollständig modular aufgebautes System, dessen Komponenten für Softwareentwicklung und eventuell Debugging auch am Personalcomputer angeschlossen werden können. Original IBM-Erweiterungskarten (Netzwerkkarten, verschiedene Schnittstellenkarten usw.) sind über eine Adapterkarte direkt auf ICB-Systemen einsetzbar.
Durch die Verwendungsmöglichkeit eines Personalcomputers zur Programmentwicklung für eine industrielle Applikation entfällt die ansonsten notwendige Anschaffung eines eigenen Entwicklungssystems (VME, ECB usw.).
Durch konsequente 19"-Technik (zum Einsatz kommen Platinen im Einfach- und/oder Doppel-Euro-Format) sind ohne großen Aufwand beliebige Schutzarten nach DIN 40 050 realisierbar.

INDUTRONIC®
Dipl.-Ing. Kreiger GmbH & Co. KG
D-8011 Poing/München
Gruberstraße 46
Telefon (0 81 21) 7 01-0
Telefax (0 81 21) 7 13 64

Ich bitte um Zusendung weiterer Informationen:

Name:

Firma:

Straße:

Plz/Ort:

Telefon:

INHALT

INFO

Ray Duncan: Wir in den USA ... 6
CeBIT 12
Compaq fertigt in Schottland 26
Taiwan greift an 30
Apples Konzept aus einem Guß 35

GRUNDLAGEN

Aufbereitung von Bildern 38

Durch Bearbeitung mit Fast-Fourier-Transformationen können Störungen in digitalisierten Bildern beseitigt werden. Die wichtigsten Algorithmen werden vorgestellt.

Arbeiten mit dem 80287 50

Die einzelnen Befehle machen deutlich, wie der Coprozessor mathematische Berechnungen wesentlich beschleunigen kann.

Der MC68030 68

In der zweiten Folge der Serie erfahren Sie, wie es um die Softwarekompatibilität des neuen Mikroprozessors von Motorola zu seinen Vorgängern wirklich steht und wie Datencache und Befehlscache arbeiten

Was Sie über Modula wissen sollten 72

Diesmal geht es um die Reservierung und die Ansprache von Speicher mit Pointern und Modulen

TEST

TextMaker 1.0 77

Für den schmalen Geldbeutel: Was die an Wordstar angelehnte Textverarbeitung für den Heim- und Hobbybereich bietet, zeigt dieser Testbericht.

Overlays unter Turbo 4.0 78

Overlays haben nach wie vor ihre Berechtigung. Borland glaubte bei Turbo-Pascal 4.0 darauf verzichten zu können. Der Overlay-Manager schließt die Lücke

Wer hat wo wann was publiziert? 79

Auf die Verwaltung von Literaturstellen spezialisiert ist Microlit. Ein Kurztest stellt die wesentlichen Merkmale dieses Programmes heraus.

Moderner PC-Assembler 80

Nicht nur ein symbolischer Debugger und vereinfachte Segmentdirektiven heben die neueste Version 5.0 des Assemblers von Microsoft von ihren Vorgängern ab.

Ein Alleskönner 84

Der neue Star LC-10 Multifont kann alles, was man von einem Matrixdrucker verlangt. Er druckt sauber und präzise und ist außerdem preiswert.

SOFTWARE

Bildschirmfunktionen für Turbo-C 87

Dialoge gehören zu den Grundforderungen komfortabler Benutzerführung. Einige abgedruckte Listings stellen für C-Programmierer ein Instrumentarium zur Verfügung.

Computergrafik: Erich Jäger

Grafik

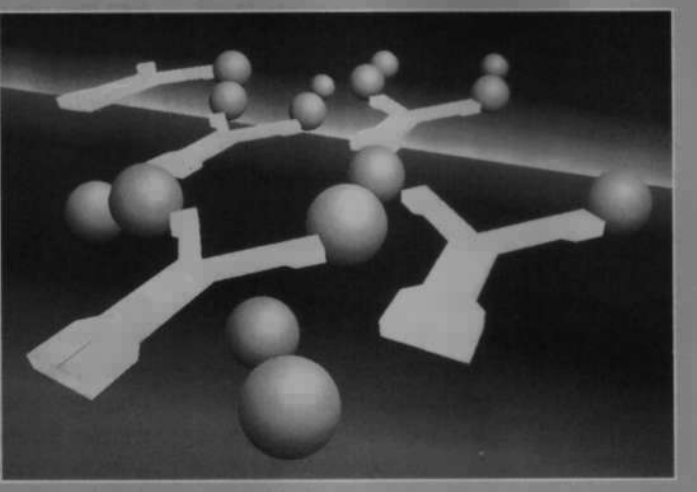
Bildinhalte sind mit Fast-Fourier-Transformationen effektiv beeinflußbar, sei es zur besseren Erkennbarkeit oder um Grafiken zu gestalten. – Manche Programme (zum Beispiel GW-Basic) verlangen eine CGA-Karte, mit der vorgestellten CGA-Emulation tut's aber auch eine Hercules-Karte. – Und weil bei Grafiken oft trigonometrische Funktionen viel Zeit kosten, lohnt sich ein kritischer Blick auf die Algorithmen von Sinus und Cosinus. – Außerdem ein Beitrag zur EGA-Karte
(Seite 114, siehe unten).

Seite 38, 94 und 105

CeBIT

Die CeBIT hat sich als eigenständige Veranstaltung den von der Gesamtmesse Hannover übernommenen Ruhm erhalten können. Nach Auskunft der Aussteller lassen sich nirgendwo so viele internationale Kontakte knüpfen wie in Hannover.
mc-Redakteur Reiner Schönrock war auf dem Messe-Gelände unterwegs und hat für Sie die wichtigsten Neuheiten zusammengestellt. Was haben Apple, Atari, Commodore, IBM und andere gezeigt? Eine ausführliche Antwort auf diese Frage bringen wir ab

Seite 12



Netzwerk

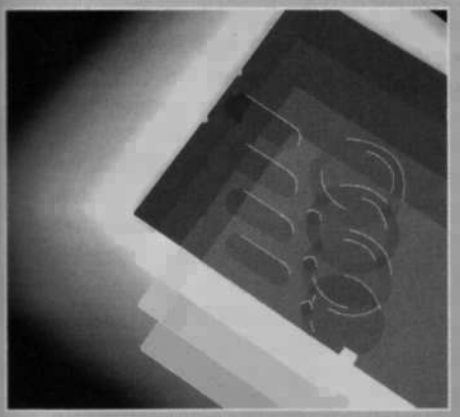
Sowohl einfache als auch recht komplexe Steuerungsaufgaben lassen sich mit dem Omega-Net verwirklichen. Mit einer Selbstbauanleitung können auch Sie preiswert alle Vorteile nutzen, die eine Vernetzung mit sich bringt. Omega-NET arbeitet mit Token-Ring-Technik.

Seite 122

EGA-Palette

Bereits 1987 hat mc ausführlich und gründlich die Theorie der EGA-Grafikkarte vorgestellt. Ihre Möglichkeiten mit den insgesamt 64 Farbtönen werden jedoch allzuoft nicht gänzlich ausgeschöpft, dazu ein Programm in Assembler zum Editieren der Farb-Palette.

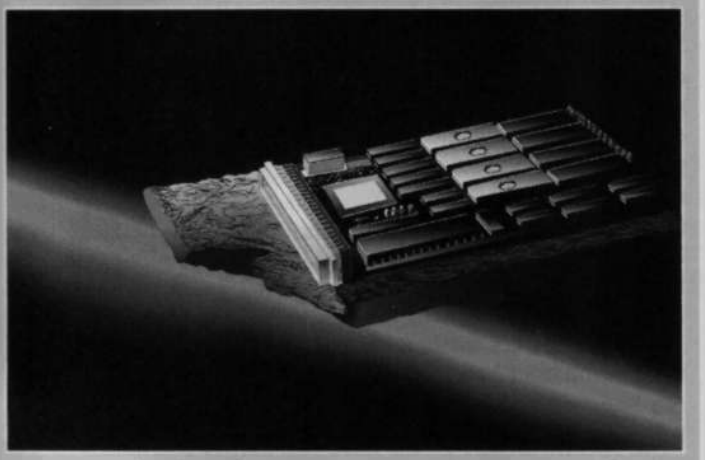
Seite 114



MASM 5.0

Beim Übergang von MASM 3.0 zu 4.0 hatte Microsoft nur wenige Details geändert. Dagegen wurde bei der Version 5.0 einiges getan. Vereinfachte Segment-Direktiven erleichtern die Handhabung, und der beigefügte Debugger Codeview gibt dem guten alten MASM ein völlig neues Gesicht. Wer in Assembler programmiert, sollte sich über den MASM 5.0 gründlich informieren.

Seite 80



Transputer

Bei einer ökonomischen Realisierung trotzdem die Universalität zu wahren, das stand für uns bei der Transputerkarte im Vordergrund. Der zweite Teil der Serie macht Sie mit den Grundlagen der Hardware vertraut und stellt die Schaltung der mc-Transputerkarte vor. Die Links sind „INMOS-kompatibel" herausgeführt, damit mehrere Karten (und auch Fremdsysteme) parallel arbeiten können. Bedient wird die Karte über die serielle Schnittstelle eines PC.

Seite 134

INHALT

Turbo-Pascal ruft 68008 91
Wie man die 68008-Karte als Coprozessor des Z80 auf dem Apple II nutzt

CGA-Emulation auf der Hercules-Karte 94
Viele GW-Basic-Benützer ärgert's: die Grafikbefehle funktionieren nicht auf einer Herculeskarte. Grund: GW-Basic setzt eine CGA-Karte voraus. Eine CGA-Emulation hilft in diesem und ähnlichen Fällen.

Schnelle Algorithmen für Sin und Cos 105
Die Berechnung von Sinus- und Cosinus-Werten nimmt relativ viel Zeit in Anspruch. Deshalb lohnt sich die Suche nach besonders schnellen Algorithmen. Ein Vergleich des im Coprozessor 8087 benutzten Verfahrens mit anderen ist sehr aufschlußreich.

Einführung in Intuition 108
Gadgets nennt Commodore die Stellen des Bildschirms, die beim Amiga von der Maus zur Auslösung einer Funktion angeklickt werden können. Mehrere Typen werden unterschiedlichen Anwendungsfällen gerecht.

Editieren der EGA-Palette 114
Weitgehend ungenutzt sind die Farbmöglichkeiten der EGA-Karte, denn meist werden die 16 darstellbaren Farben so belassen, wie man sie vom Textmodus her kennt. Mit 64 sieht die Welt schon wesentlich bunter aus.

Vorführeffekt ade 116
Trotz aller Planung kommen peinliche Pannen bei Software-Demonstrationen immer wieder vor. Ein Programm, das bestimmte Tastatureingaben vormerkt und auch wiederholen kann, ist da sehr willkommen.

C-Funktionen im Amiga-ROM 118
Im Amiga-ROM befinden sich nützliche C-Routinen (zum Beispiel Umwandlungen zwischen verschiedenen Zahlentypen). Sie lassen sich auch aus Assemblerprogrammen heraus nutzen und sparen somit Entwicklungsarbeit.

HARDWARE

Omega-Net – Ein Peripherie-Netz für PCs 122
Mit modernster Token-Ring-Technik für Steuerungsaufgaben

Die mc-Transputerkarte 134
Im zweiten Teil der Serie steht die Technik im Vordergrund.

RUBRIKEN

Editorial 3
Impressum 6
Briefe 10
Info 12
Bücher 36
Spruch des Monats 76
Markt 148
Verlagsinformationen 181
Vorschau 182

mc
Die Mikrocomputer-Zeitschrift

Franzis-Verlag GmbH
Karlstraße 37–41, 8000 München 2
Postfach 37 01 20, 8000 München 37
Telefon (0 89) 51 17-1
Telex 5 22 301
Telefax (0 89) 51 17-3 79
Tedas (0 89) 59 84 23 und 59 64 22

Chefredakteur:
Dipl.-Math. Ulrich Rohde

Stellvertretender Chefredakteur:
Dipl.-Ing. (FH) Ulrich Kruppe

Redaktion:
Redaktionsassistentin:
Rita Schleser (0 89) 51 17-3 54
Ressortredakteure: Horst Brand, Dipl.-Ing. (FH) Wolfgang Hascher (fl), Aktuelles: Reiner Schönrock, PC-Technik: Dipl.-Ing. (FH) Dieter Strauß
Anschrift der Verantwortlichen siehe Verlagsanschrift

Korrespondent: USA:
Ray Duncan

Ständiger Mitarbeiter:
Dipl.-Ing. (FH) Herwig Feichtinger, Helga M. Schmidt (zu erreichen unter der Anschrift der Redaktion)

Art Direktion:
Dipl.-Designer (FH)
Volker Hilbel

Layout, Grafik, Herstellung:
Josef Wurzinger

Franzis-Labor:
Neucom electronic GmbH

Software Service:
SHAMROCK Software Vertrieb
(0 89) 59 54 68

Verantwortlich für den Inhalt:
Dipl.-Math. Ulrich Rohde

Verantwortlich für Anzeigen:
Dietger Kötter

Gesamtherstellung:
Franzis-Druck GmbH
Karlstraße 35
8000 München 2
Telefon (0 89) 51 17-1

© 1988 für alle Beiträge bei Franzis-Verlag GmbH

ISSN: 0720-4442

Vertriebskennzeichen:
B 7745 E

Sonderdrucke:
Jakob Wintersberger

Lizenzen:
Georg Geschke, Königstr. 8,
3000 Hannover 1,
Telefon (05 11) 34 53 62

Urheberrechte:
Die in mc veröffentlichten Beiträge sind urheberrechtlich geschützt. Alle Rechte, insbesondere das der Übersetzung in fremde Sprachen, vorbehalten. Auch die Rechte der Wiedergabe durch Vortrag, Funk- oder Fernsehsendung im Magnettonverfahren oder ähnlichem Wege bleiben vorbehalten. Fotokopien für den persönlichen oder sonstigen eigenen Gebrauch dürfen nur von einzelnen Beiträgen oder Teilen daraus als Einzelkopien hergestellt werden.

RAY DUNCAN

Wir in den USA ...

In nur eineinhalb Jahren sind hier die PCs mit 80386 im Bauch zum Liebling eines jeden „Power Users", jeden Programmierers und jeden Fachautors geworden. Nachdem der 80386-Chip mehr und mehr auf dem Markt verfügbar wurde, schossen die Verkaufszahlen der 80386-PCs in den Himmel und die Preise purzelten in dieser kurzen Zeit um die 50 %. All dieses Getue um den 80386 mag Sie verwundern, wenn Sie noch keine Zeit fanden, sich mit diesem Prozessor und seinen einzigartigen Eigenschaften zu beschäftigen. Aufgrund der Erfahrungen mit Intels 80286 könnte man schließlich vom 80386 nicht mehr erwarten, als daß man sich für ein bißchen mehr Geschwindigkeit einige Kompatibilitätsprobleme einhandelt.

Der 80386 ist aber sehr viel mehr als ein schneller 286: Er ist ein echter 32-Bit-Super-Mikroprozessor mit einem enormen, virtuellen Adreßraum, mit Cache-Speicher-Einrichtung und einer Befehls-Pipeline. Sein Befehlssatz ist sehr viel mächtiger, effizienter und orthogonaler, als der seiner Vorgänger 8086 bis 80286, sobald er im echten 386-Modus läuft. Aber das Wichtigste: Der 80386 besitzt Vorkehrungen, damit er aufwärtskompatibel bleibt und das erleichtert das Leben derer sehr, die Betriebssysteme bauen oder Anwendungen entwickeln.

Beim Start und nach Reset beginnt der 80386 im sogenannten Real Mode zu arbeiten. Darin emuliert er im Wesentlichen den 8086, wie es auch der 80286 tut. Im Real Mode kann der 80386 nur 1 MByte adressieren und seine Mechanismen für den Schutz von Task- und User-Speicherbereichen gegen illegale Zugriffswünsche sowie die Mechanismen für die virtuelle Speicherverwaltung sind abgeschaltet. Die Segmentregister enthalten wie gewohnt die physikalischen Paragraph-Adressen. In diesem Modus läuft MS-DOS mit allen seinen Anwenderprogrammen auf dem Prozessor, wobei der Durchsatz im Schnitt zwei- bis dreimal größer ist.

Wenn bestimmte Tabellen und Deskriptoren initialisiert worden sind, kann der 80386 per Programm in den Schutz-Modus umgeschaltet werden. Einmal in diesem Modus, wird ein von der Hardware überwachtes Privileg-Ebenen-System wirksam, wobei einige Befehle nur für die höchste Privileg-Ebene reserviert sind. Außerdem wird die Behandlung von Adressen geändert: Es wird alles eine gewisse Stufe indirekter. Der Wert in einem Segmentregister ist im Schutzmodus ein Selektor, ein Zeiger auf eine Deskriptortabelle, die selbst die Basisadresse und die Länge eines physikalischen Speichersegmentes enthält sowie die dazu gehörenden Attribute (read-only, read-write, executable) und Privileg-Informationen. Immer wenn von einem Programm aus der Speicher angesprochen werden soll, wird von der Hardware auf dem Prozessor die physikalische Adresse anhand des gerade wirksamen Deskriptors gebildet und gleichzeitig überprüft, ob ein gültiger Speicherzugriff vorliegt.

Insofern scheint der Schutzmodus des 80386 nicht von dem des 80286 verschieden zu sein. Es gibt aber höchst bedeutsame Unterschiede. Zunächst kann ein einziges Segment beim 80386 bis zu 4 GByte groß werden. Da läßt der 80286 nur noch von Ferne grüßen – mit seinen maximal 64 KByte großen Segmenten, die auch die Größe des maximalen, zusammenhängend adressierbaren Speicherblockes darstellen. Schon allein diese Eigenschaft war ein Grund, daß sehr viele Leute die Ankunft des 80386 erleichtert feierten, denn nun konnten große, segmentüberschreitende Programme unter einem linearem Speichermodell geradeaus geschrieben werden. Man kann jetzt einfach ein einziges sehr großes Segment definieren, alle Segmentregister des Prozesses auf dessen Basisadresse richten und fortan die ganze Segmentierung vergessen.

Darüber hinaus gibt es beim 80386 einen weitere Stufe im Mechanismus zur Adreßbildung, das Paging. Sobald Paging aktiviert ist, können die von der Hardware mit Selektor-Register und Offset interpretierten linearen Adressen zusätzlich fortlaufend in Speicherblöcke von je 4 KByte Größe abgebildet werden. Die dafür notwendigen Tabellen müssen durch ein Betriebssystem im physikalischen Speicher aufrecht erhalten werden. Diese Zerlegung des virtuellen Adreßraumes in Blöcke fester Größe ist für Memory Manager, die Programme und Daten zwischen Massenspeicher und Hauptspeicher je nach Bedarf hin und herbewegen, äußerst bequem (bedenken Sie, daß ein Speichersegment jede Größe, von 1 Byte bis zum Maximum, annehmen kann, was für das Speicher-Management im virtuellen Betrieb eine harte Nuß würde).

Schließlich gibt es beim 80386 zwei verschiedene Varianten des Schutzmodus. Er kann einmal im 16-Bit-Modus eine logische Obermenge der 80286-Befehle absolvieren (und so 16-Bit-Programme im Schutzmodus fahren) und einmal in seinem eigenen 32-Bit-Schutzmodus laufen. Dort gibt es 32-Bit-Befehle, 32-Bit-Register und zum Beispiel Stackoperationen ohne „Strafzeiten". Ein einziges Bit in einer Task-Deskriptor-Tabelle steuert die Umschaltung dieser zwei Modi. Das heißt, daß ein Betriebssystem zwischen zwei Applikationen, die jeweils verschiedene Schutz-Modi verlangen, ohne irgendeinen zusätzlichen Aufwand umschalten kann.

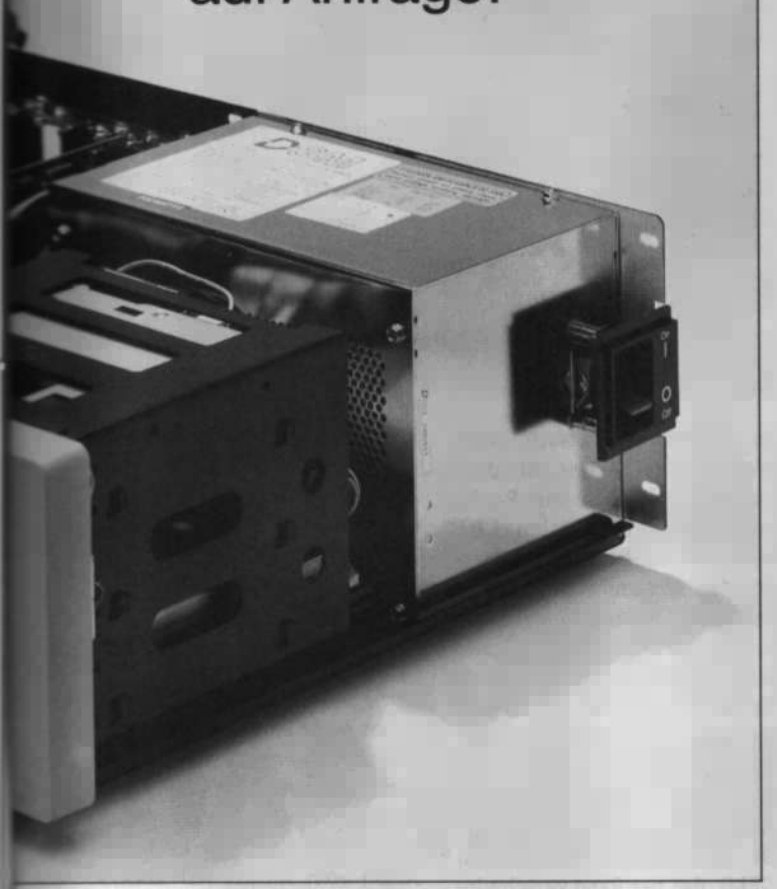
Am faszinierendsten ist allerdings beim 80386 sein virtueller 8086-Modus. Ein Betriebssystem für den Schutzmodus kann beim 80386 bis zu 1 Mbyte große Segmente definieren, die speziell für 8086-Real-Modus-Programme reserviert sind. Solche Programme sehen alles so, als würden sie auf einem 8086 ablaufen. Ein jedes solches Segment ist eine virtuelle 8086-Maschine mit eige-

(Bitte lesen Sie weiter auf Seite 10.)

m Modular-Konzept

NEU!

OEM-Konditionen
auf Anfrage!



Bei den hier aufgeführten Karten handelt es sich um qualitativ hochwertige Module, die auch in herkömmlichen IBM- oder IBM-kompatiblen Systemen einsetzbar sind. Auf alle Komponenten ½ Garantie! Alle Preise inclusive 14 % MwSt.
IBM ist eingetragenes Warenzeichen der IBM Corporation.

❶ **DSM AT286-modular-Fertiggerät** bestehend aus AT-modular-Board V 2.0, 512 KB RAM (100 ns), BUS-Platine, Festplatten-Floppy-Controller, Monochrome-Graphik-Karte, stabiles Gehäuse, 200 Watt Netzteil, MF-2-kompatible Tastatur (102 Tasten), 5¼" 1.2 MB Floppy-Disk-Laufwerk, 20 MB Seagate-Festplatte (65 ms), 14"-Monitor. Mit MS-DOS 3.3®, GW-Basic, und deutschen Handbüchern. Komplett aufgebaut und geprüft (48 Std. Burn-In), 1 Jahr Garantie
Best.-Nr. M0203 **DM 4.198,–**

❷ **DSM AT286-modular-Kit 2**
Wie Pos. 1 als Teile-Kit
Best.-Nr. M0202 **DM 3.798,–**

❸ **DSM AT286-modular-Kit 1**
Wie Pos. 2 ohne Hard-Disk, ohne MS-DOS und Monitor
Best.-Nr. M0201 **DM 2.798,–**

❹ **AT286-modular-Board V2.0**
auf 1 MB RAM ausbaubar, mit 512 KB Speicherbausteinen bestückt, 7 Kanal DMA, 16 Interrupt-Ebenen, 3 programmierbare Timer, 1 Realtime-Clock, CMOS-RAM, bis 64 KB ROM, Keyboard-Controller, 10 MHz, 0-Wait! Neuer CHIPS-TECHNOLOGY-Satz und mit Standard-Speicherbausteinen bestückbar. Award-BIOS. Inkl. Handbuch
Best.-Nr. M0102, 100ns RAMS's **DM 1.298,–**
Best.-Nr. M0100, 120ns RAM's **DM 1.198,–**
Best.-Nr. M0101, ohne RAM's **DM 898,–**

❺ **AT386-modular-Board**
Hochleistungs-CPU-Karte mit 80386 Prozessor (32-Bit) 16 MHz. Bis zu 16 Mbyte-0-Wait-Speicher, Sockel für 80287 und 80387 Coprozessor
Best.-Nr. M0142 **DM 3.798,–**

❻ **Bus-Platine**
mit insgesamt 8 Steckplätzen, davon 6 AT-Slots sowie 2 XT-Slots
Best.-Nr. M0103 **DM 98,–**

❼ **Festplatten-Floppy-Controller**
Western Digital®-Chip-Satz. Verwaltet bis zu 4 Laufwerke, 2 Floppy-Laufwerke (360 KB / 1.2 MB) und 2 Hard-Disk-Laufwerke. Optional 1 zusätzliches Floppy-Laufwerk
Best.-Nr. M0129 **DM 398,–**

❽ **Monochrom-Grafik-Karte**
HERCULES™-kompatibel, mit parallelem Druckeranschluß
Best.-Nr. M0107 **DM 148,–**

❾ **EGA-Standard-Grafik-Karte EGA** 640x350, 640x200, 320x200, CGA, MDA, 16/64 Farben
Best.-Nr. M0108 **DM 398,–**

❿ **SUPER-EGA Level 5**
800x600, 640x480, 640x350, 640x200, 320x200, 16/64 Farben
Best.-Nr. M0111 **DM 598,–**

⓫ **RAM-Karte**
2 MegaByte Extended und Expanded Memory, 10 MHz, (0 KB-RAM) AT-RAM-Erweiterungs-Platine
Best.-Nr. M0112 **DM 398,–**

⓬ **Gehäuse**
Stabile Ausführung mit Befestigungsmaterial
Best.-Nr. M0122 **DM 198,–**

⓭ **Netzteil**
200 W, kurzschlußfest incl. Netzkabel
Best.-Nr. M0128 **DM 238,–**

⓮ **Tastatur**
MF-2-kompatibel, 102 Tasten
Best.-Nr. M0126 Deutsch **DM 298,–**
Best.-Nr. M0127 ASCII **DM 298,–**

⓯ **3½" 1.44 MB Floppy-Disk-Laufwerk**
OS/2®-Format (2 MB unform.)
Best.-Nr. M0104 **DM 398,–**

⓰ **3½" 1.2 MB Floppy-Disk-Laufwerk**
AT-kompatibel (1.6 MB unform.)
Hersteller: TEAC/NEC
Best.-Nr. M0105 **DM 298,–**

⓱ **5¼" 1.2 MB Floppy-Disk-Laufwerk**
AT-kompatibel (1.6 MB unform.)
Hersteller: NEC
Best.-Nr. M0106 **DM 298,–**

⓲ **Festplatten-Laufwerke**
a) 20 MB Seagate (65 ms)
Best.-Nr. M0123 **DM 598,–**
b) 40 MB Seagate
Best.-Nr. M0124 **DM 1.150,–**
c) 80 MB Seagate (28 ms)
Best.-Nr. M0125 **DM 2.498,–**

⓳ **Coprozessoren**
80287-1 (10 MHz)
Best.-Nr. M0118 **DM 698,–**
80387 (16 MHz)
Best.-Nr. M0119 **DM 1.398,–**

⓴ **Monitore**
14" Monitore mit Schwenkfuß
a) grün
Best.-Nr. M0134 **DM 325,–**
b) bernstein
Best.-Nr. M0135 **DM 329,–**
c) **EGA-Farb-Monitor**
Auflösung bis 640x350 Punkte
Best.-Nr. M0136 **DM 998,–**
d) **EIZO-Farb-Monitor**
Auflösung bis 820x620 Punkte für den Betrieb aller EGA-Karten, TTL-Eingang, umschaltbar auf ANALOG-Eingang
Best.-Nr. M0137 **DM 1.798,–**

㉑ **Betriebssystem MS-DOS 3.3®** deutsch mit GW-BASIC und deutschen Handbüchern
Best.-Nr. M0120 **DM 298,–**

㉒ **Zubehör**
a) Handbuch AT-modular-Computer im praktischen Kunststoffordner
Best.-Nr. M0130 **DM 39,–**
b) Kabelsatz (2x Floppy und 1x Hard-Disk)
Best.-Nr. M0131 **DM 49,–**

Fordern Sie unsere ausführlichen Listen (PC-News und PC-Software-News mit über 500 Programmen) an!

Händleranfragen erwünscht. (Bitte mit Händlernachweis)

DSM Digital Service GmbH

Zentrale: Landwehrstraße 37 · 8000 München 2
Filiale: Schillerstraße 17 · 8000 München 2

DSM Tel. 089/596916
Telex 523545 dsm d · Fax 089/598261
Computer Systeme
Ein Unternehmensbereich der *DSM Digital* Service GmbH

nem Adreßraum, eigenem I/O-Port-Adreßraum und eigener Interrupt-Vektor-Tabelle. Es können viele virtuelle 8086-Maschinen gleichzeitig laufen, wobei jede sich so fühlt, als hätte sie einen kompletten Computer für sich. Der entscheidende Unterschied zwischen dem virtuellen 8086-Betrieb und dem Real Mode für 8086: Im virtuellen Modus gibt es die vielen Schutzmechanismen, die Privileg-Prüfung und die Mechanismen für den virtuellen Speicherbetrieb auch dann, wenn ein 8086-Prozess läuft. Damit kann ein 8086-Programm auf einer virtuellen 8086-Maschine nicht mit dem Betriebssystem in Konflikt geraten und auch nicht andere Prozesse stören. Greift ein solches Programm auf Speicher zu, der nicht zu seiner virtuellen Maschine gehört, oder will es I/O-Adressen nutzen, zu welchen es keine Zugangsberechtigung besitzt, dann wird eine „Exception" generiert (ein bestimmter Hardware-Interrupt) und das übergeordnete Betriebssystem schaut nach dem Rechten. Dessen Exception Handler kann nun zwischen Prozeß-Abbruch, Übernahme der Programmausführung unter eigener Regie oder einer anderen geeigneten Alternative wählen. Mit dem virtuellen 8086-Betrieb ist die Tür zur Multitasking-Welt für bestehende DOS-Programme geöffnet worden, und zwar bei bestehenden Schutzmechanismen. Es ist zu erwarten, daß es in Kürze Betriebssysteme gibt, die nebeneinander MS-DOS-Programme, 16-Bit-Programme für den Schutz-Modus und 32-Bit-Tasks (ebenfalls im Schutz-Modus) laufen lassen können – bei voller Wahrung der Integrität der Konkurrenten. Wie das aussieht, deutet heute schon Windows/386 an.

Ein anderer bedeutender Fortschritt beim 80386 gegenüber dem 80286 ist seine schnelle Modus-Umschaltung. Als der 80286 konstruiert wurde, beherrschte noch CP/M die Welt, MS-DOS steckte noch in den Kinderschuhen und die Ingenieure bei Intel hätten sich nicht träumen lassen, daß möglicherweise der Durchbruch eines 80286-Betriebssystemes (für den Schutz-Modus) daran hängen könnte, daß es auch möglich wäre, darunter auch 8086-Real-Modus-Programme einhängen zu können. Deshalb implementierten die Intel-Ingenieure beim 80286 zwar einen „Schalter", um vom Real-Modus in den Schutz-Modus zu gelangen, aber keinen Weg zurück, ohne daß man den Prozessor anhält.

Daß es beim 80286 keinen vernünftigen Weg gab, um vom Schutzmodus wieder zurück in den Real-Modus zu kommen, das war für die OS/2-Entwickler ein sehr großes Hindernis. Denn es war eins der wichtigen Entwicklungsziele bei OS/2, daß man mit „alten" MS-DOS-Anwendungen im Multitasking parallel zu neuen Programmen für den Schutz-Modus fahren kann. Nach langem Suchen fanden die Entwickler zum Schluß eine Besonderheit in der Architektur des 80286, die dazu mißbraucht werden konnte, einen Reset auszulösen, der für ihre Zwecke schnell genug war (eine Methode, die Gordon Letwin, einer der Chef-Entwickler von OS/2, wie folgt charakterisierte: „Stopping the car to change gears"*). Außerdem fanden sich die Designer von OS/2 gezwungen, eine undokumentierte 80286-Instruktion auszunutzen (LOADALL), um im Real-Modus auf den erweiterten Speicher zugreifen zu können, damit die Anzahl der Modus-Umschaltungen minimiert werden konnte.

Der 80386 wurde dagegen konstruiert, als schon klar war, daß unbedingt auch die Aufwärts-Kompatibilität mit dem 8086-Real-Modus benötigt wurde. Daher unterstützt der 80386 schnelle Modus-Umschaltung auch von der Software her. Dazu muß man nur die geeigneten Bits im Maschinen-Status-Wort editieren. Die im Handel erhältliche OS/2-Version überprüft, ob sie etwa auf einem 80386 läuft und nutzt dessen schnelles Schalten bei Bedarf aus. Das reduziert den Overhead beim Kontext-Umschalten für Real-Modus-Programme und verkürzt die Interrupt-Antwortzeit.

* Das Auto zum Schalten anhalten.

BRIEFE

AT-Motherboard

In mc 3/88, Seite 16 wirbt die Firma VLSI mit einem Motherboard der IC Serie VL 82 CPAT-QC, deren Steckplätze auch zusätzlich AT-kompatibel sind. Das ist für Endkunden, die Ihre Motherboards im XT tauschen wollen, ein großer Anreiz und ich frage daher an, wie die amerikanische oder besser deutsche Adresse von VLSI lautet und ob es evtl. Händler gibt, die Motherboards mit diesem Chipsatz von VLSI vertreiben (die Probleme mit der Tastatur sind bekannt, ich kann umschalten).
Für Ihre Mitteilung wäre ich sehr dankbar, es würde sicher viele Leser interessieren.

Manfred Buchholz,
8044 Unterschleißheim

Anmerkung der Red.: Die von Ihnen gesuchte Adresse lautet:
VLSI Technology,
Rosenkavalierplatz 10
8000 München 81
☎ 0 89/92 69 05-0

Tempus-Patch

Es ist vollbracht! Nach einer Woche harter Arbeit mit dem Debugger fand ich den Patch, der Tempus der Version 1.01 veranlaßt, auch mit dem Blitter-TOS so zusammenzuarbeiten, daß das Desktop wieder erreicht wird: Mit einem Diskmonitor wird der vierte Sektor von Tempus gesucht. In diesem Sektor wird das Byte der sektorrelativen Adresse 5FH vom alten Wert 08 auf den neuen Wert von 07 geändert – fertig!
Der alte Fehler (vielleicht im GEM), daß der Rand verschwindet, tritt aber auch hier noch auf.

Bodo Wenzel,
3300 Braunschweig

Modular programmieren

Kaum jemand kann es sich verkneifen, Seitenhiebe auf das gute alte Basic auszuteilen. So auch in der ersten Folge der Serie „Modular programmieren" (mc 2/88), wo das Beispiel ggT „viele" GOTOs enthält, nämlich zwei Stück. Dabei ist dieses Beispiel schlecht gewählt. Denn schon das Microsoft Basic unter CP/M kannte die Schleifenkonstruktion mit WHILE / WEND. Damit unterscheidet sich dann das Basic-Programm im wesentlichen nicht mehr von denen in Pascal oder Modula.
Schon seit vielen Jahren gibt es in Basic Programmelemente, die die strukturierte Programmierung unterstützen und die Abhängigkeit von Zeilennummern lockern. Ich denke dabei z. B. an das hervorragende Basic der HP-Tischrechnerserie 200. Wer in einem modernen Dialekt, z. B. Turbo Basic, unstrukturiert programmiert, ist selber schuld.
Der wirklich gravierende Unterschied zwischen Basic und Pascal oder Modula besteht in den Datenstrukturen. In Basic muß nichts vorab deklariert werden; dafür stehen nur die einfachsten Variablentypen zur Verfügung. Pascal und ähnliche Sprachen leisten hier selbstverständlich unvergleichlich mehr.

Leo Michels,
Untereisesheim

Anmerkung der Red.: Ihr Brief steht stellvertretend für eine ganze Reihe von Leserbriefen gleichen Inhalts. Anscheinend ist die im betreffenden Beitrag enthaltene Aussage, daß moderne Basic-Dialekte ein erhebliches Mehr an Strukturelementen aufweisen, nicht deutlich genug geworden. Hier ging es jedoch um die Tatsache, daß sich Modula 2 der Problematik schon vom Ansatz her anders nähert – und so war es unserer Meinung nach passend, es mit einem Basic in seiner ursprünglichen Definition zu vergleichen, die eben kein WHILE, WEND usw. kennt.
Was die Datenstrukturen angeht: Außer den KI-Spezialsprachen Lisp und Prolog gibt es wohl keine verbreitete Programmiersprache, die Zeichenketten unbestimmter Länge so komfortabel behandelt wie gerade Basic ...

GANZE INFORMATIONEN SIND DAS HALBE GESCHÄFT

Wirtschaftlicher Erfolg hängt immer mehr davon ab, daß Sie aus der Flut von Informationen die für Ihr Unternehmen wichtigen herausfiltern, bewerten und für Ihre Entscheidungen nutzen.

Nur auf der Basis abgesicherter Informationen lassen sich heute die Entscheidungen treffen, durch die das Unternehmen morgen die nötigen Gewinne machen soll.

Im Rahmen der finanziellen Betreuung sehen wir daher auch in der Rationalisierung der Zahlungsvorgänge eine wichtige Aufgabe für die Zusammenarbeit mit unseren Kunden.

Das entlastet Sie von zeitraubenden Verwaltungsarbeiten.

Sprechen Sie also mit uns, wenn es um die Verbesserung von Gelddispositionen, um belegloses Datenträgeraustausch, Scheck- und Wechseleinzug, Lohn- und Gehaltsabrechnungen geht.

Damit Ihnen die Zeit bleibt, Unternehmer zu sein.

wenn's um Geld geht – Sparkasse

Dr. Klaus Dieter Brinkmann, Dr. Hans-Werner Eisermann

Aufbereitung von Bildern

Quantisierung, algorithmische Bearbeitung und Visualisierung von bildhaften Daten (Teil 2)

Die FFT (Fast Fourier Transform) und ihre mathematischen Grundlagen werden behandelt. Mit ihr können Störungen in Bildern weitgehend behoben werden. Die wichtigsten Algorithmen zur Bildverarbeitung sind in einem Programmpaket in Fortran zusammengefaßt.

Funktionen reeller Variablen lassen sich oft in Form von Reihen darstellen. Dabei spielen die nach dem französischen Ägyptologen und Naturwissenschaftler J. Fourier benannten Reihen eine herausragende Rolle, insbesondere wegen ihrer weiten Anwendungsbereiche in Technik und angewandter Forschung. In der einfachsten Form besteht die Reihe aus Sinus- und Cosinustermen, die mit Koeffizienten versehen sind. In *Bild 1* wird anhand eines einfachen Beispiels gezeigt, wie eine Funktion schon durch Addition weniger Sinus- und Cosinusschwingungen recht gut angenähert werden kann.

Diese anschauliche Darstellung durch die bekannten trigonometrischen Funktionen wird leider völlig unübersichtlich, wenn mit Fourierreihen gerechnet werden soll. Dann zeigt es sich, daß es sehr viel einfacher ist, für die Berechnungen komplexwertige Exponentialschwingungen zu verwenden. Das sieht auf den ersten Blick komplizierter aus, ist es aber nicht. Im folgenden wird deshalb nur die Exponential-Schreibweise verwendet; der grundsätzliche Zusammenhang ist gegeben durch die Euler-Identität:

$\exp(i \times f \times t) = \cos(f \times t) + i \times \sin(f \times t)$.

Auf weitere Beziehungen zwischen Sinus/Cosinus- und Exponentialdarstellung wird hier nicht eingegangen; man findet viele Umrechnungen in den Formelsammlungen. Während die Fourierreihen

$$f_p(t) = \sum_{n=-N}^{N} c_n e^{i2\pi nft} \qquad (1)$$

nur zur Darstellung periodischer Funktionen geeignet sind, kann die Fouriertransformation auch bei nichtperiodischen Vorgängen angewendet werden. Die Transformationsformeln gewinnt man aus denen für die Fourierreihen dadurch, daß man die Periodendauer gegen Unendlich streben läßt. Damit gilt für eine Funktion h(t):

$$H(f) = \int_{-\infty}^{\infty} h(t)\, e^{-i2\pi ft}\, dt \qquad (2)$$

$$h(t) = \int_{-\infty}^{\infty} H(f)\, e^{-i2\pi ft}\, df \qquad (3)$$

Kennt man beispielsweise den Verlauf einer Funktion h in Abhängigkeit von der Zeit t, kann man die Frequenzen f berechnen, die den Verlauf der zeitlichen Funktion ausmachen – man bestimmt das Frequenzspektrum (Fourieranalyse). Mit (3) ist auch der umgekehrte Weg, die Konstruktion einer Funktion der Zeit aus ihrem Frequenzspektrum möglich (Fouriersynthese). Diese Symmetrie wird offensichtlich, wenn man h in die Formel von H einsetzt: man erhält die Identität. h ist die Transformation von H und umgekehrt.

Fouriertransformation mit dem Digitalrechner

Die Fouriertransformation besteht im allgemeinen, wie aus (2) und (3) ersichtlich, aus einem reellen und einem imaginären Teil. Man assoziiert damit Betrag (Amplitude) und Phase der Spektrallinien. H(f) und h(t) sind kontinuierliche Funktionen. Für Anwendungen in der digitalen und damit diskreten Rechnerwelt ist diese Form der Fouriertransformation ungeeignet. Man verwendet die diskrete Fouriertransformation

$$F_m = \sum_{n=0}^{N-1} f_n\, e^{-i2\pi mn/N};\ m = 0 \ldots N-1 \qquad 4$$

mit der inversen Transformation

$$f_n = 1/N \sum_{m=0}^{N-1} F_m\, e^{i2\pi mn/N};\ n = 0 \ldots N-1 \qquad (5)$$

Anwendung von (4) auf eine Funktion, die beispielsweise einen zeitlichen Verlauf kennzeichnet, liefert die komplexen Spektrallinien F_m, die zu den diskreten Frequenzen $f = m/N \times t$ gehören (man vergleiche mit Formel (1)). Der Faktor 1/N in (4) dient zur Normierung. Je dichter man die „Meßwerte" f_m legt, desto besser wird die tatsächliche Funktion approximiert. Üblicherweise spricht man nicht von Meßwerten, sondern von Stützstellen für die Approximation. Sind viele Stützstellen bekannt, können die Frequenzen, die auftreten, gut bestimmt werden. Die Anzahl N der Stützwerte bestimmt die Auflösung.

Um mit dieser Methode bei der digitalen Bildverarbeitung eine brauchbare Auflösung zu erzielen, ist der Rechenaufwand selbst für Hochleistungsrechner zu groß; denn da Bilder zweidimensionale Objekte sind, ist die Fouriertransformation zweimal anzuwenden: die Anzahl der komplexen Multiplikationen und Additionen steigt quadratisch. Grund: Die Anzahl der Stützstellen – das sind in der digitalen Bildverarbeitung die Bildpunkte – nimmt quadratisch zu. Für ein Bild 256 × 256 (= 64K Pixel) wären allein 4 Millionen dieser zeitaufwendigen Fließkomma-Operationen durchzuführen. Wir in 1 ms durchschnittlich eine Operation ausgeführt, wie bei Hochleistungsrechnern der Fall, dann dauert allein die reine Rechenzeit für die Transformation eines Bildes über eine Stunde. Mit der Entwicklung der Signalverarbeitung entstand der dringende Bedarf für eine schnelle Methode, Fouriertransformationen durchzuführen.

Der Begriff FFT (Fast Fourier Transform, schnelle Fourier Transformation) wurde zum Schlagwort. Es wurden (und werden) viele Algorithmen entwickelt, die für bestimmte Anwendungen optimiert sind. Kriterien sind Schnelligkeit, Einfachheit, Speicherplatzverwendung, Verarbeitung nichtquadratischer Bilder usw. Bekannt wurde der FFT-Algorithmus, den Tukey und Cooley 1965 veröffentlichten [2]. In bezug auf Geschwindigkeit und Einfachheit zeigt er recht günstige Eigenschaften. Die meisten heutigen Realisierungen basieren auf diesem Verfahren.

Wichtig ist die Feststellung, daß die FFT keine Näherung zur diskreten FT ist, sondern ebenso genaue Ergebnisse liefert wie sie. Die Erhöhung der Rechengeschwindigkeit erkaufte man sich im wesentlichen dadurch, daß die Anzahl der Stützwerte bzw. Bildpunkte eine Zweierpotenz sein muß. Dadurch wird die Anzahl der Operationen von N^2 auf $N \times 1d(N)$ reduziert. Konkret bedeutet dies z. B. bei einer Auflösung von 512 × 512 eine Einsparung von $512^2 = 262\,144$ auf $512 \times 9 = 4608$, also mehr als 250 000 Operationen. Neben der einfacheren Speicherorganisation ist das ein Grund dafür, daß die meisten Bildverarbeitungssysteme für Bilder der Größe 256 × 256, 512 × 512 oder 1024 × 1024 ausgelegt sind.

Bevor man den Tukey-Cooley-Algorithmus anwenden kann, müssen die zu transformierenden Werte in eine bestimmte Reihenfolge gebracht werden. Aus der Theorie geht hervor, daß die Indizes in einer baumartigen Struktur vertauscht werden müssen. Zum Glück ist dafür nicht eine komplizierte Berechnung erforderlich; vielmehr leistet die Umkehrung der Bits der Indexzahlen genau das Gewünschte. Aus diesem Grund wird die spezielle Sortierung der Indizes als Bit-Reverse-Shuffling bezeichnet. *Tabelle 1* zeigt einige Indizes und deren bit-umgekehrte Werte.

Da Bilder als zweidimensionale „Funktionen" aufgefaßt werden, muß auch die FFT zweimal angewendet werden, einmal auf die Zeilen, einmal auf die Spalten. Um nicht auch das Bit-Reverse-Shuffling zweimal vorzunehmen, werden beide Umsortierungen zusammengefaßt; dies leistet die Routine FFBRS2 (*Bild 7*). Die zur Umsortierung benötigten Vertauschungsindizes werden mit der Subroutine FFTAB dynamisch berechnet, die bei der Transformation benötigten Sinus- und Cosinuswerte in der Subroutine FFSIN. Alle diese häufig verwendeten Werte werden in Form einer Tabelle zur Verfügung gestellt; das spart Zeit. Legt man sich auf eine bestimmte Bildgröße fest, dann ist es natürlich zweckmäßig, die Tabellen für Vertauschungsindizes und Winkelfunktionen nicht dynamisch zu berechnen, sondern in einer Initialisierungsroutine die Tabellen einzulesen – so kann noch weiter Zeit gespart werden. Nach diesen Vorbereitungen wird die zweidimensionale FFT durchgeführt. Zur Vermeidung einer IF-Abfrage in der Verarbeitungsschleife werden die Subroutinen für Hin- und Rücktransformation getrennt aufgerufen: FFT2HX und FFT2RX. Im Falle der Rücktransformation erfolgt in der Routine noch die Normierung gemäß (5).

Die Darstellung der transformierten Bilder

Die Anordnung der Werte aus dem Tukey-Cooley-Algorithmus eignet sich zwar gut für die Berechnungen, ist aber unzweckmäßig für die Darstellung und die Interpretation. Die Routine FFDISP hilft diesem Zustand ab: sie bereitet die Zahlenmatrix so auf, daß die Punktsymmetrie der fouriertransformierten Funktion deutlich wird. FFNORM ist die dazu inverse Routine; mit ihr wird das Bild wieder in die für die Rücktransformation günstige Form gebracht. Bei beiden Subroutinen wird nicht der Pixelinhalt verändert, sondern die Pixel werden nur anders angeordnet.

Dadurch wird diesen Bildern ein anderes Koordinatensystem zugrundegelegt. Man zählt die Bildpunkte nicht mehr von 1...N, sondern von –v...v–1 und von –u...u–1. So wird die Punktsymmetrie um (0,0) deutlich. Das ist wichtig, weil anders eine anschauliche Interpretation des Fourierspektrums nicht möglich ist. Auf wichtige Symmetrieeigenschaften der Fouriertransformation sei hier hingewiesen:

Periodizität:
$F(u,v) = F(u+N,v) = F(u,v+N) = F(u+N,v+N)$

Symmetrie:
$F(u,v) = F^*(-u,-v)$, was dazu führt, daß $|F(u,v)| = |F(-u,-v)|$.
* bedeutet konjugiert komplex.

Die Darstellung des Ergebnisses der Fouriertransformation eines Bildes erfolgt indem das „Betragsbild" aus Real- und Imaginärteil abgebildet wird; man erhält so ein Maß für die Energie des Bildes. Die Interpretation der Einzelbilder – Realbild (Amplitudenspektrum) und Imaginärbild (Phasenspektrum) – ist noch nicht so hinreichend erforscht, daß sie Aussagekraft besitzen. Da die Spektralwerte sehr stark mit zunehmender Frequenz abnehmen, wird üblicherweise eine logarithmische Darstellungsform

$$D(u,v) = \log(1+|F(u,v)|)$$

verwendet. So können die Charakteristika der FFT besser auf dem Bildschirm erkannt werden; vom mathematischen Standpunkt aus ist diese Umrechnung überflüssig. Man beachte, daß die Nullwerte des Spektrums erhalten bleiben; negative Werte können nicht auftreten. Auf dem Bildschirm wird das FFT-Spektrum als Grauwertbild ausgegeben; die Konvention ist, daß große Spektralwerte hell, kleine Spektralwerte dunkel dargestellt werden. Dabei wird eine Normierung der Grauwerte vorgenommen, so daß die darzustellenden Werte nur den

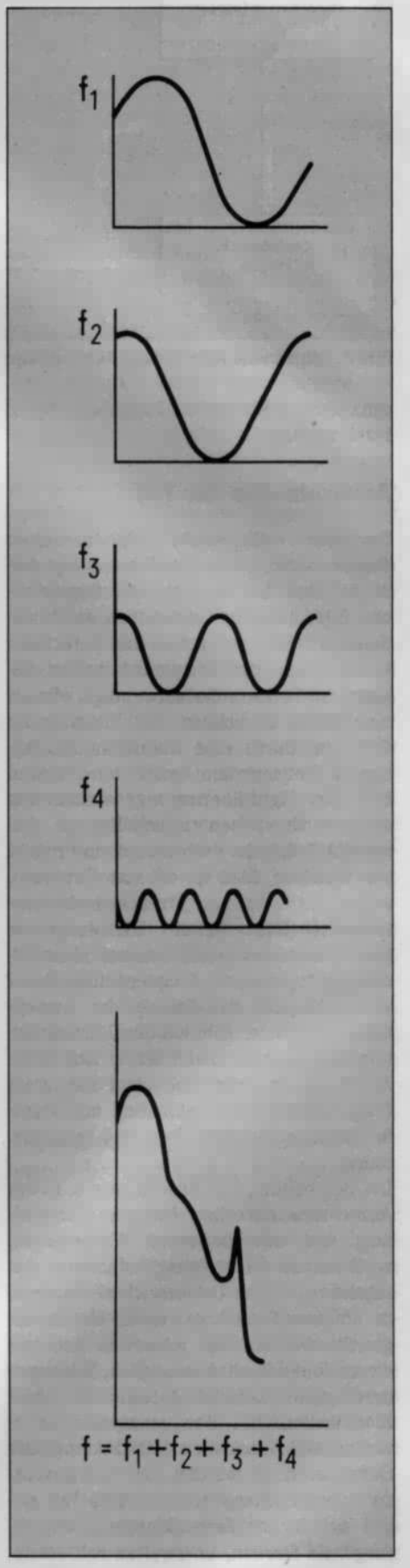


Bild 1. Durch Addition von Sinus- und Cosinusschwingungen können Funktionen angenähert werden

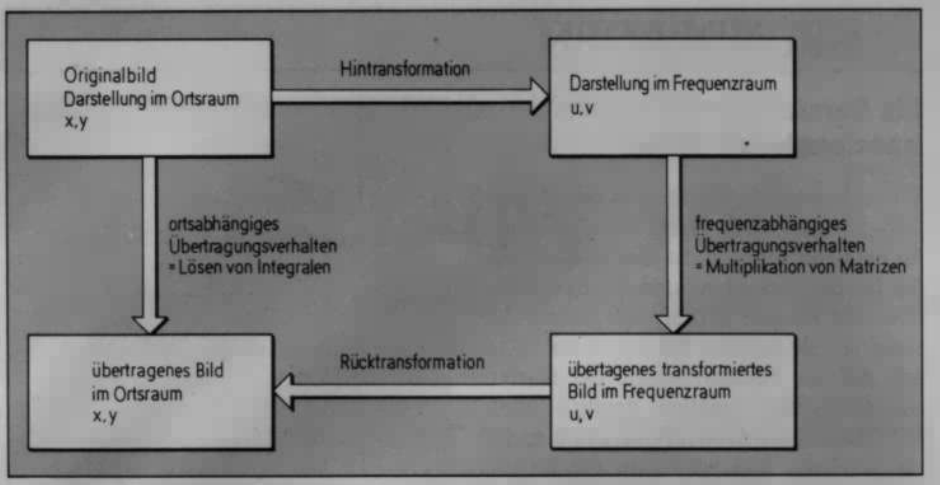


Bild 2. Prinzipielle Anwendung der Fouriertransformation in der Signalübertragung

ganzzahligen Bereich 0...255, also 1 Byte/Pixel, umfassen.

Anwendungen der FFT

Der Grund dafür, solche rechenintensiven Transformationen durchzuführen, liegt darin, daß man das Verhalten von Signalen – und Bilddaten sind letztendlich nichts anderes – relativ gut kennt und berechnen kann, wenn das Frequenzverhalten bekannt ist. So kann das aufwendige, oftmals unmögliche Berechnen der Integrale im Ortsraum durch eine Matrizenmultiplikation im Frequenzraum ersetzt werden. Sind bei der Signalübertragung/-verarbeitung mehrere Komponenten beteiligt, so „beschränkt" sich das Problem auf ein Produkt von Matrizen. Man spricht vom Ortsraum, wenn man sich auf die Pixel des nichttransformierten Bildes bezieht. Die Menge der Pixel des fouriertransformierten Bildes bilden den sogenannten Frequenzraum. *Bild 2* veranschaulicht das Prinzip der Anwendung der FFT im Rahmen der Bildverarbeitung und zeigt die Gleichheit zwischen der Arbeit im Ortsraum und einer Arbeit im Frequenzraum in Kombination mit Transformationen zwischen Orts- und Frequenzraum.

Die Bearbeitung von Bildern, wie z. B. das Aufnehmen mit einer Kamera, Übermittlung und anschließendes Wiedergeben, wird durch Übertragungsfunktionen beschrieben, die im Ortsbereich als Integral, im Frequenzbereich als Fourier-Matrix dargestellt werden. Bei teilweiser Kenntnis dieser Funktion wird es möglich, Störungen durch mathematische Operationen – hier Modifikation des Frequenzspektrums – weitgehend zu beheben. Verwacklung oder Defokussierung werden im Fourierspektrum leicht erkennbar: im ersten Fall zeigen sich in der Verwacklungsrichtung orthogonale Streifen, im zweiten Fall entstehen konzentrische Kreise um den Mittelpunkt. Durch Manipulationen im Frequenzbereich, also mit Hilfe der Fouriertransformation, können solche Störungen einigermaßen gut behoben werden. Im Ortsbereich wäre es um ein Vielfaches aufwendiger.

Als anschauliches Beispiel für Operationen im Frequenzraum kann eine Filterung mit einem Tiefpaß dienen, der die hohen Frequenzanteile dämpft oder abschneidet. Während im Ortsraum hierfür eine Integration erforderlich ist, müssen im Frequenzraum lediglich die entsprechenden Frequenzen mit einem Faktor zwischen 0 und 1 gewichtet werden. Solche Modifikationen kann man im Fourierspektrum auch interaktiv vornehmen, indem man bestimmte Frequenzbereiche Null setzt. Realisiert wird dies durch Nullsetzen der entsprechenden Bereiche im Speicher. Bei der Veränderung der Werte ist natürlich darauf zu achten, daß tatsächlich die Zahlenwerte in der fouriertransformierten Bildmatrix, und nicht nur die Grauwerte des Bildwiedergabespeichers, verändert werden. Das Ergebnis der Rücktransformation entspricht dann einer Tiefpaßfilterung. Auf diese Weise können Bilder verändert werden: es entstehen Verfremdungen oder Bildverbesserungen.

Bild 3 zeigt das Original und das fouriertransformierte Bild. Mit der Störung von *Bild 4* wird das Original zu *Bild 5* überlagert. Dadurch erhält das Fourierspektrum eine andere Struktur, die jedoch das Grundmuster des Originals enthält, wie in *Bild 6* dargestellt. Durch Nullsetzen und Abschwächen verschiedener Frequenzen und anschließender Rücktransformation wird der störende Eindruck vermindert, wie in *Bild 6* gezeigt. Die Manipulation hat allerdings auch den Effekt, daß das Bild unschärfer wirkt. Durch verfeinerte Wahl der Nullstellen kann die Störung weitgehend beseitigt werden.

Programm-Implementation

Der große Speicherplatzbedarf zwingt zu Kompromissen bei der Programmierung der FFT, da die Adressierung der Speicherplätze sichergestellt sein muß. Wird – wie in dieser Arbeit – unter dem Betriebssystem MS-DOS gearbeitet, welches für Datenbereiche im allgemeinen eine Maximalgröße von 64 KByte vorsieht, so bieten sich folgende Lösungen an:

(1) nur kleinere Bilder bearbeiten

(2) einen Fortran-Compiler verwenden, der für alle Feld-Elemente die Segment- und Offset-Adresse generiert (Large Data Model bei Microsoft)

(3) das Bild so unterteilen, daß mit Files auf der RAM-Disk gearbeitet werden kann.

Zum Lösungsweg (3) seien einige kurze Hinweise gegeben. Um das Problem zu umgehen, nur 64K ohne besonderen Offset adressieren zu können, müssen kleinere Felder bearbeitet werden. Das erreicht man am einfachsten durch eine Abfrage des Index. Überschreitet der Index bestimmte Grenzen, wird der bereits bearbeitete Teil des Bildes weggeschrieben und der neu zu bearbeitende Teil eingelesen. Das ist ein gut praktikables Verfahren, solange Ein- und Ausgabe nicht zu oft durchgeführt werden müssen.

Das ist hier bei der FFT-Berechnung der Zeilen der Fall. Obwohl zusätzliche Rechenzeit durch die Abprüfung des Index nötig ist – bei einer Viertelung des Bildes müssen über einmillionenmal die Prüffunktion aufgerufen und eine Integer×4-Abfrage durchgeführt werden – wird die Ausführung der FFT nicht untragbar verlängert. Anders bei der Berechnung der Spalten. Wird das beschriebene Verfahren auch hierfür einfach übernommen, muß zu jeder Berechnung mehrmals ein- und ausgegeben werden. Abhilfe wird geschaffen durch Vertauschung von Zeilen und Spalten des teiltransformierten Bildes. Damit wird der Speicher so belegt, daß das häufige Ein- und Ausgeben entfallen kann: die Verarbeitung kann ebenso vorgenommen werden wie beim ersten Teil der Transformation.

Die Symmetrieeigenschaft sowohl des reellen als auch des imaginären Teils können dazu verwendet werden, das Bild platzsparend – nämlich auf die Hälfte reduziert – abzuspeichern. Da für den Realteil vier Werte mehr als N/2 abzuspeichern sind, für das Imaginärbild aber stets vier Pixel den Wert 0 haben, werden diese Speicherplätze zur Speicherung von Realwerten genutzt. Die Konsequenz ist, daß Real- und Imaginärteil immer gemeinsam abgelegt werden müssen. Die Subroutine FFPAC nutzt die Symmetrieeigenschaft aus und speichert das fouriertransformierte Bild mit halbem Speicherbedarf ab, FFUNP dekomprimiert die Daten in das für die Anwendung der FFT günstige Format. Bedingt durch die Arithmetik ist die Symmetrie

Bild 3. Das Originalbild (links) und die Fouriertransformierte des Originalbildes (rechts)

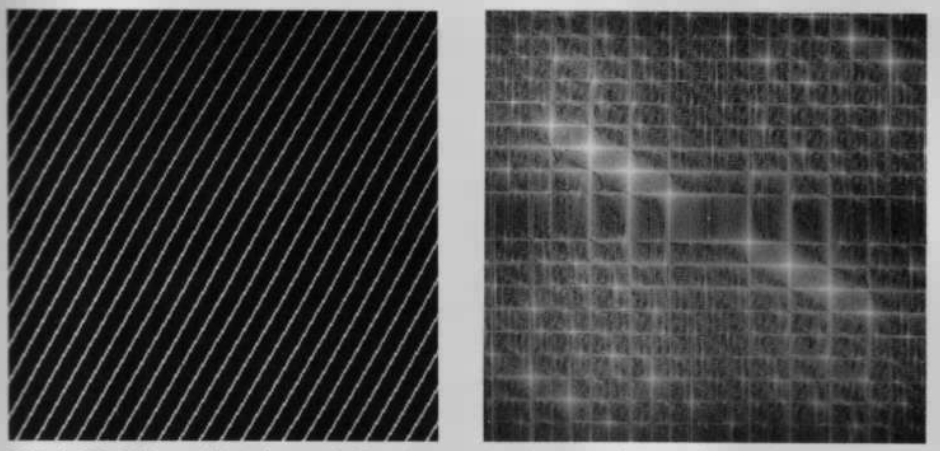


Bild 4. Die Störung (links) und die Fouriertransformierte der Störung (rechts)

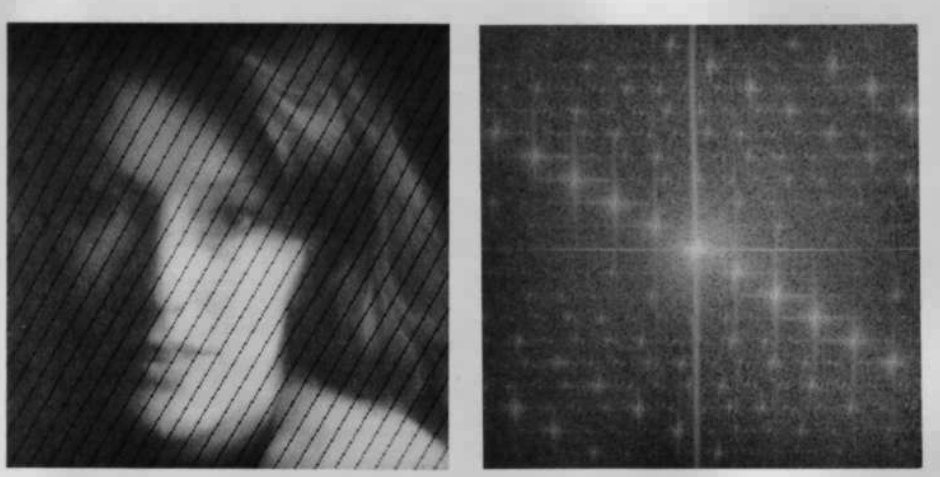


Bild 5. Das Originalbild mit der Störung überlagert (links) und die Fouriertransformierte des überlagerten Bildes (rechts)

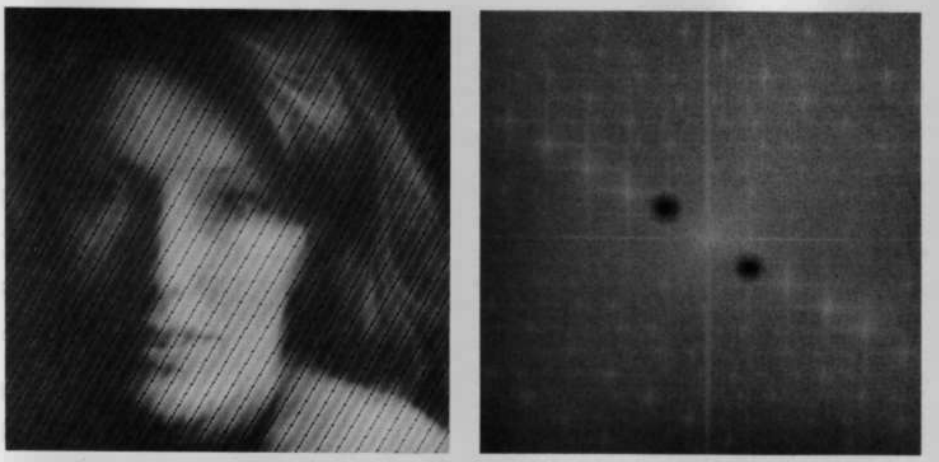


Bild 6. Das gefilterte, rücktransformierte Bild (links) und dessen Fouriertransformierte (rechts), bei dem Frequenzbereiche abgeschwächt oder ganz unterdrückt werden

fouriertransformierter Bilder nicht ganz exakt. Den Unterschied bemerkt man, wenn man nach der Anwendung von FFPAC und FFUNP – dadurch wird eine Symmetrie erzwungen – das „originale" fouriertransformierte Bild subtrahiert.

Das Programm-System PCBILD dient in erster Linie als Hilfsmittel zum Verständnis der Einfachheit und Übersichtlichkeit von Algorithmen der Bildverarbeitung. Es wurde deshalb nicht immer unbedingt Wert gelegt auf optimale Programmierung mit Ausnutzung aller Tricks und Raffinessen der Hard- und Software. Vielmehr wurde eine modulare, leicht modifizierbare und weitestgehend sprachunabhängige Programmstruktur gewählt. Da allen Informatik-Theoretikern zum Trotz Fortran noch die am weitesten verbreitete Programmiersprache ist, soll hier die Implementierung in dieser Sprache vorgestellt werden.

Anzumerken ist hierzu, daß den Verfassern bei Beginn der Arbeiten nur der Compiler von Professional zur Verfügung stand, der von „Big Blue" vertrieben wird. Zu diesem Produkt wird in der Einleitung erklärt, daß es sich um die „Interpretation der Norm" durch die 3-Buchstaben-Firma handelt.

Bei der praktischen Arbeit merkt man sehr schnell, daß diese Interpretation oft umständlich und schwer nachvollziehbar ist. Die Kompatibilität von Programmen, die durch andere Compiler einwandfrei bearbeitet werden, ist oft nicht gewährleistet, so daß manche Programmteile unter Professional umständlich umgeschrieben werden müssen. Deshalb sollten die Leser etwaige Schwerfälligkeiten nicht den Verfassern anlasten, die bei manchen Programmteilen auf unverständliche Fehlermeldung stießen und viele graue Haare bekamen, sondern dem verfügbaren Compiler.

Datenkonzept, Parametervereinbarung und Schnittstelle

PCBILD unterscheidet zwischen Integer und Real-Bildern, wobei letztere das Ergebnis von mathematischen Transformationen sind. Obwohl üblicherweise die Integer-Bilder nur Werte zwischen 0 und 255 annehmen und somit durch 8 Bit beschreibbar sind, muß wegen des Compilers die Deklaration als INTEGER×2 erfolgen (direkte Zuweisung an Byte-Daten ist nicht möglich). PCBILD stellt dem Benutzer zwei Integer-Bilder BILD1 und BILD2 gleichzeitig im Hauptspeicher zur Verfügung. Zur Speicherung von Real-Bildern wird ein Bild REBILD vom Typ REAL×4 zur Verfügung gestellt.

Wie im *Bild 8* gezeigt, wird dieses Bild durch EQUIVALENCE den Bildern BILD1

```
c  Test - Programm fuer FFT
c
c  Die Routinen fuer Einlesen und Schreiben sowie fuer
c  Darstellung der Bilder sind fuer den betreffenden Rechner
c  zu erstellen.
c
c
      parameter (line=256)
c
c
c  Tabellen
c
      integer   vtab(line)
      real*4    stab(line),ctab(line)
c
c  Bildspeicher
c
      real*4    ar(line*line),ai(line*line)
      real*4    dr(line*line),di(line*line)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      n=line
c
c  Berechnen der Tabellen
c
      call fftab (vtab,n)
      call ffsin (stab,ctab,n)
c
c
c  Einlesen der Bilddaten
c
      call readp (ar,ai,n)
c
c  Bitreverse-Shuffling
c
      call ffbrs2 (ai,n,vtab)
      call ffbrs2 (ar,n,vtab)
c
c
c  Hintransformation
c
      call fft2hx (ar,ai,n,stab,ctab)
c
c
c  in Display-faehiges Format bringen
c
      call ffdisp (ar,dr,n)
      call ffdisp (ai,di,n)
c
c
c  Ausgabe auf Bildschirm etc.
c
      call dispp (ar,ai,n)
c
c
c  wieder in Verarbeitungsformat bringen
c
      call ffnorm (dr,ar,n)
      call ffnorm (di,ai,n)
c
c  Bitreverse-Shuffling
c
      call ffbrs2 (ar,n,vtab)
      call ffbrs2 (ai,n,vtab)
c
c
c  Ruecktransformation
c
      call fft2rx (ar,ai,n,stab,ctab)
c
c  ruecktransformiertes Bild auf Disk schreiben
c
      call writep (ar,ai,n)
c
      stop
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c  Die Ein-/Ausgabe-Routinen muessen jeweils an die Hardware angepasst
c  werden
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine dispp (ar,ai,n)
c  dummy
      return
      end
      subroutine writep (ar,ai,n)
c  dummy
      return
      end
      subroutine readp (ar,ai,n)
c  dummy
      return
      end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine fft2hx (cr,ci,n,stab,ctab)
c
c  Zweck: Routine zur zweidimensionalen schnellen Fourier Transformation
c         von xy nach uv
c
c  cr:   Realteil Ein- und Ausgabe
c  ci:   Imaginaerteil Ein- und Ausgabe
c  n:    Anzahl der Stuetzwerte (cr, ci enthalten je n**2 Werte)
c  stab: Sinus Tabelle
c  ctab: Cosinus-Tabelle
c
c  Die Bereiche cr und ci muessen bit-reverse sortiert sein.
c  Der Imaginaerbereich ci wird mit 0 besetzt angenommen.
c
      implicit integer*4 (i-n)
      integer*4 i,j,k,l,ioff
      real      cr(1),ci(1),stab(1),ctab(1)
c
c
c ---------------------- bearbeitung der zeilen ----------------------
c
c
c  transformation
      do 199 ilin=1,n
        ioff=(ilin-1)*n
        l=1
        isc=1
111     if (l.ge.n) go to 119
        istep=l+l
        do 115 m=1,l
          wr=ctab(isc)
          wi=stab(isc)
          isc=isc+1
          k=m
113         i=k+ioff
            j=i+l
            crtmp=wr*cr(j)-wi*ci(j)
            citmp=wr*ci(j)+wi*cr(j)
            cr(j)=cr(i)-crtmp
            ci(j)=ci(i)-citmp
            cr(i)=cr(i)+crtmp
            ci(i)=ci(i)+citmp
            k=k+istep
          if (k.le.n) go to 113
115     continue
        l=istep
        go to 111
119     continue
c
199   continue
c
c
c ---------------------- bearbeitung der spalten ----------------------
c
c
c  transformation
      do 299 icol=1,n
```

Bild 7. Programme zur Fouriertransformation

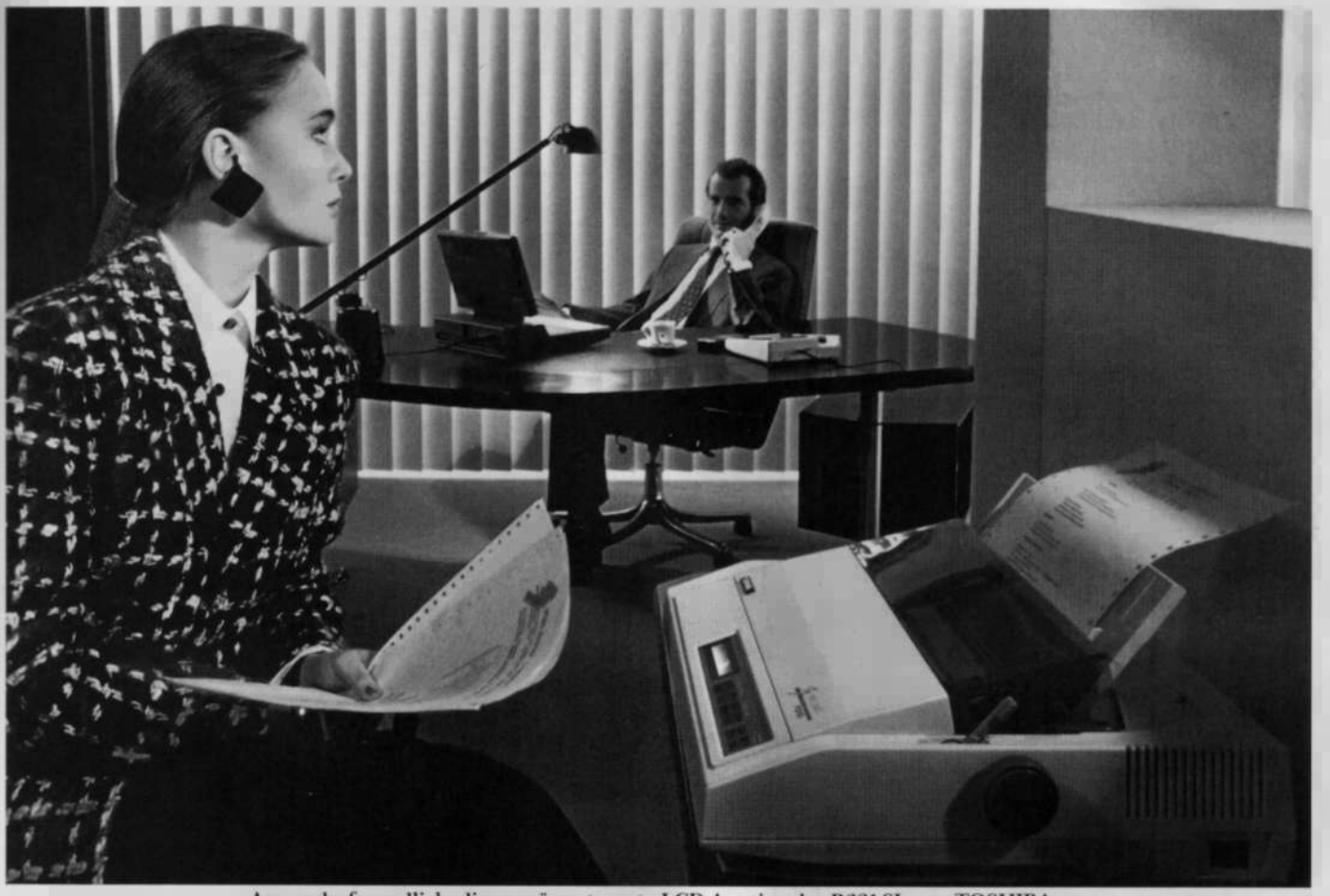
```
        l=1
        isc=1
211     if (l.ge.n) go to 217
        nl=n*l
        istep=l+l
c
        do 215 m=1,l
          wr=ctab(isc)
          wi=stab(isc)
          isc=isc+1
          k=m
213         i=(k-1)*n+icol
            j=i+nl
            crtmp=wr*cr(j)-wi*ci(j)
            citmp=wr*ci(j)+wi*cr(j)
            cr(j)=cr(i)-crtmp
            ci(j)=ci(i)-citmp
            cr(i)=cr(i)+crtmp
            ci(i)=ci(i)+citmp
            k=k+istep
          if (k.le.n) go to 213
215     continue
        l=istep
        go to 211
c
217   continue
c
299   continue
c
c  exit
      return
      end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      SUBROUTINE FFT2RX (CR,CI,N,STAB,CTAB)
c
c  Zweck: Routine zur zweidimensionalen schnellen Fourier Transformation
c         von uv nach xy
c  cr:   Realteil Ein- und Ausgabe
c  ci:   Imaginaerteil Ein- und Ausgabe
c  n:    Anzahl der Stuetzwerte (d.h. es existieren n**2 werte fuer cr und ci)
c  stab: Sinus-Tabelle
c  ctab: Cosinus-Tabelle
c
c  Die Bereiche cr und ci muessen bit-reverse sortiert sein.
c
      implicit integer*4 (i-n)
      integer*4 i,j,k,l,ioff
      real    cr(1),ci(1),stab(1),ctab(1)
c
c ---------------------- bearbeitung der zeilen ----------------------
c
c  transformation
      do 199 ilin=1,n
        ioff=(ilin-1)*n
        l=1
        isc=1
111     if (l.ge.n) go to 119
        istep=l+l
        do 115 m=1,l
          wr=ctab(isc)
          wi=-stab(isc)
          isc=isc+1
          k=m
113         i=k+ioff
            j=i+l
            crtmp=wr*cr(j)-wi*ci(j)
            citmp=wr*ci(j)+wi*cr(j)
            cr(j)=cr(i)-crtmp
            ci(j)=ci(i)-citmp
            cr(i)=cr(i)+crtmp
            ci(i)=ci(i)+citmp
            k=k+istep
          if (k.le.n) go to 113
115     continue
        l=istep
        go to 111
119     continue
c
199   continue
c
c
c ---------------------- bearbeitung der spalten ----------------------
c
c
c  transformation
      do 299 icol=1,n
        l=1
        isc=1
211     if (l.ge.n) go to 217
        nl=n*l
        istep=l+l
c
        do 215 m=1,l
          wr=ctab(isc)
          wi=-stab(isc)
          isc=isc+1
          k=m
213         i=(k-1)*n+icol
            j=i+nl
            crtmp=wr*cr(j)-wi*ci(j)
            citmp=wr*ci(j)+wi*cr(j)
            cr(j)=cr(i)-crtmp
            ci(j)=ci(i)-citmp
            cr(i)=cr(i)+crtmp
            ci(i)=ci(i)+citmp
            k=k+istep
          if (k.le.n) go to 213
215     continue
        l=istep
        go to 211
c
217   continue
c
299   continue
c
c
c -------------------- normieren der zeitwerte -------------------
c
      nn=n*n
      cnn=nn
      do 301 i=1,nn
      cr(i)=cr(i)/cnn
      ci(i)=ci(i)/cnn
301   continue
c
c
c  exit
      return
      end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      SUBROUTINE FFBRS2 (CC,N,VTAB)
c
c  Zweck: Routine fuer zweidimensionales Bitreverse-shuffling per Tabelle
c
c  cc:   Bildbuffer (Real- oder Imaginaerteil), Input und Output
c  n:    Anzahl der Spalten (=Anzahl der Zeilen), Input
c  vtab: Tabelle der eindimensionalen Vertauschungsindizes, Input
c
c  Um schnelleres Umsortieren zu erreichen, werden die Realwerte in
c  dieser Subroutine als Integer*4 Werte behandelt.
c
c
      integer   vtab(1)
      integer*4 cc(1),cctmp
      integer*4 i,j,l
c
      i=0
      do 109 ilin=1,n
      l=(vtab(ilin)-1)*n
        do 109 icol=1,n
        j=vtab(icol)+l
        i=i+1
```

```
          if (i.ge.j) go to 109
          cctmp=cc(i)
          cc(i)=cc(j)
          cc(j)=cctmp
109     continue
        return
        end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        subroutine fftab (table,n)
c
c  Zweck: berechnet eindimensionale Vertauschungstabelle (bit-reverse)
c
c         table: Tabelle der Vertauschungsindizes
c                Output, n Real*4 Speicherplaetze
c         n:     Anzahl Zeilen, Input, Integer
c
c
        implicit integer*4 (i-n)
        integer table(1)
c
c
        do 120 i=1,n
120     table(i)=i
        n1=n-1
        j=0
        do 130 i=1,n1
          l=n/2
121       jl=j+l
          if (jl.le.n1) go to 123
          l=l/2
          go to 121
123       jmodl=(j-((j/l)*l))
          j=jmodl+l
          if (j.le.i) go to 130
          i1=i+1
          j1=j+1
          table(i1)=j1
          table(j1)=i1
130     continue
c
        return
        end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        SUBROUTINE FFSIN (STAB,CTAB,N)
c
c  Zweck: Routine zur Berechnung der zur FFT verwendeten Sinus-
c         und Cosinus Tabellen
c
c
        integer   n,i,l,lstep,m
        real*4    stab(1),ctab(1),pi,a
c
        data pi/3.14159/
c
c
        i=1
        l=1
103     continue
        if (i.ge.n) go to 199
        lstep=l+l
        do 105 m=1,l
          a=-pi*(m-1)/l
          stab(i)=sin(a)
          ctab(i)=cos(a)
          i=i+1
105     continue
        l=lstep
        go to 103
c
c  exit
199     continue
        return
        end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        SUBROUTINE FFPAC (BBUFR, BBUFI, CBUF, N)
c
c  Zweck: Die in der Bearbeitungsform abgelegten reellen und imaginaeren
c         FFT Daten werden auf halben Speicherbedarf komprimiert.
c
c  Eingabe: BBUFR(N*N): reelle FFT Daten
c           BBUFI(N*N): imaginaere FFT Daten
c           N:          Anzahl der Spalten (=#Zeilen =Zweierpotenz)
c
c  Ausgabe: CBUF(N*N): komprimierte FFT Werte
c
c  Hinweis: CBUF darf sich nicht mit einem anderen Buffer ueberlappen!
c
c
        implicit integer*4 (i-n)
        integer*4 bbufi(1),bbufr(1), cbuf(1)
c
c
        nn2=n*n/2
        n2=n/2
c
        j=n2
        k=nn2
        do 101 i=1,nn2
        j=j+1
        cbuf(i)=bbufr(j)
        k=k+1
        cbuf(k)=bbufi(j)
101     continue
c
        k=nn2+1
        cbuf(k)=bbufr(1)
        i=nn-n2+1
        j=nn2+n2+1
        cbuf(i)=bbufr(j)
c
        return
        end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        SUBROUTINE FFUNP (CBUF, BBUFR, BBUFI, N)
c
c  Zweck: Die auf halben Speicherbedarf komprimierten Daten der FFT
c         werden dekomprimiert. Die Anordnung der Daten geschieht in
c         der fuer Bearbeitung besonders geeigneten Form.
c
c  Eingabe: CBUF(N*N):  Buffer der komprimierten FFT Daten
c           N:          Anzahl der Zeilen (=#Spalten =Zweierpotenz)
c
c  Ausgabe: BBUFR(N*N): Buffer der dekomprimierten reellen FFT Werte
c           BBUFI(N*N): dto. imaginaere FFT Werte
c
c  Hinweis: CBUF darf nicht mit einem anderen Buffer ueberlappen!
c           Aus Zeitgruenden wird Integer- und Floating Format gemixt.
c
c
        implicit integer*4 (i-n)
        integer*4 cbuf(1),bbufr(1),isav
        real*4    bbufi(1),xsav
c
        equivalence (xsav,isav)
c
c
        nn=n*n
        nn2=n*n/2
        n2=n/2
        n21=n2-1
c
        j=n2
        k=nn2
        do 101 i=1,nn2
        j=j+1
        bbufr(j)=cbuf(i)
        k=k+1
        isav=cbuf(k)
        bbufi(j)=-xsav
```

Die Drucker von TOSHIBA.

Ghostwriter.

Leise, in einem Arbeitsgang, gestochen scharfe Schönschrift und exakt auflösende Grafikbilder zu produzieren und kompatibel zu allen gängigen Systemen zu sein, das sind die herausragenden Eigenschaften der Drucker von TOSHIBA. Mit eingebautem Traktor, Einzelblattzuführung, acht verschiedenen Schriftarten mit je über 100 Varianten bieten die Drucker von TOSHIBA alles, was ein moderner Drucker leisten sollte.



Anwenderfreundlich, die menü-gesteuerte LCD-Anzeige des P321SL von TOSHIBA.

BERATUNGSGUTSCHEIN

P321SL

P341SL

P351/2

Hier sehen Sie alle TOSHIBA Drucker auf einen Blick.
Um genau zu wissen, welcher der Richtige ist, sollten Sie sich beraten lassen.

Wenn Sie Informationsmaterial über die Drucker von TOSHIBA haben wollen, schicken Sie den ausgefüllten Beratungsgutschein bitte an TOSHIBA.
Oder wenden Sie sich direkt an einen autorisierten TOSHIBA Fachhändler, dort können Sie die Drucker auch im Original sehen.

Firma/Branche

Name

Straße

PLZ Ort

Telefon

MC 588/321

In Touch with Tomorrow

TOSHIBA

KOPIERER • COMPUTER • DRUCKER • FERNKOPIERER

TOSHIBA Europa (I.E.) GmbH, Bereich Deutschland, Abteilung Marketing IPS, Hammer Landstraße 115, 4040 Neuss.

```
101     continue
c
        iv1=2
        jv1=n
        jv2=nn2+n2
        iv2=jv2+2
        ih1=nn2+n2+1
        jh1=ih1
        ih2=nn2+1
        jh2=ih2
c
        do 105 k=1,n21
        bbufr(iv1)=bbufr(jv1)
        bbufi(iv1)=-bbufi(jv1)
        iv1=iv1+1
        jv1=jv1-1
        bbufr(iv2)=bbufr(jv2)
        bbufi(iv2)=-bbufi(jv2)
        iv2=iv2+1
        jv2=jv2-1
        ih1=ih1+n
        jh1=jh1-n
        bbufr(ih1)=bbufr(jh1)
        bbufi(ih1)=-bbufi(jh1)
        ih2=ih2+n
        jh2=jh2-n
        bbufr(ih2)=bbufr(jh2)
        bbufi(ih2)=-bbufi(jh2)
105     continue
c
        i=nn
        k1=n+2
c
109     continue
        k2=k1+n2-2
        do 111 k=k1,k2
        bbufr(i)=bbufr(k)
        bbufi(i)=-bbufi(k)
        i=i-1
111     continue
c
        i=i-1
        k1=k1+n2
        if (k2.lt.nn2) go to 109
        j=nn2+1
        bbufr(1)=cbuf(j)
        j=nn-n2+1
        i=nn2+n2+1
        bbufr(i)=cbuf(j)
        bbufi(1)=0.
        j=n2+1
        bbufi(j)=0.
        j=nn2+1
        bbufi(j)=0.
        j=j+n2
        bbufi(j)=0.
c
c
c
        return
        end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        SUBROUTINE FFDISP (INBUF,OUTBUF,N)
c
c  Zweck: Dient zur Umsortierung der FFT Werte aus dem fuer Berechnungen
c         geeigneten Format in die symmetrische Anordnung, die fuer die
c         Darstellung am Bildschirm ueblich ist. Routine mu- fuer reelle
c         und imaginaere Werte getrennt aufgerufen werden.
c
c  Eingabe: INBUF(N*N): Buffer der reellen oder imaginaeren FFT Werte
c                       im Berechnungsformat
c           N:          Anzahl der Spalten (=#Zeilen =Zweierpotenz)
c
c  Ausgabe: OUTBUF(N*N): Buffer der reellen oder imaginaeren FFT Werte
c                        im symmetrischen Format
c
c  Hinweis: INBUF und OUTBUF duerfen nicht ueberlappen!
c
c
        implicit integer*4 (i-n)
        integer*4 inbuf(1),outbuf(1)
c
        k=0
        n2=n/2
        do 101 i=1,n
        icol=i+n2
        if (icol.gt.n) icol=icol-n
        do 101 j=1,n
        k=k+1
        ilin=j+n2
        if (ilin.gt.n) ilin=ilin-n
        l=(ilin-1)*n+icol
        outbuf(l)=inbuf(k)
101     continue
c
        return
        end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        SUBROUTINE FFNORM (INBUF, OUTBUF, N)
c
c  Zweck: Dient zur Umsortierung von der symmetrischen Anordnung der
c         FFT Werte auf Berechnungsformat. Routine mu- faer reelle und
c         imaginaeren Teil getrennt aufgerufen werden.
c
c  Eingabe: INBUF(N*N): Buffer fuer reelle oder imaginaere symmetrisch
c                       angeordnete FFT Werte
c           N:          Anzahl der Spalten (=#Zeilen =Zweierpotenz)
c
c  Ausgabe: OUTBUF(N*N): Buffer fuer reelle oder imaginaere FFT Werte
c                        in der fuer Berechnungen zweckmae-igen Anordnung
c
c  Hinweis: INBUF und OUTBUF duerfen nicht ueberlappen!
c
c
        implicit integer*4 (i-n)
        integer*4 inbuf(1),outbuf(1)
c
        nn=n*n
        n2=n/2
        nn2=nn/2
c
        i1=1
        i2=i1+n2
        i3=nn2+1
        i4=i3+n2
c
        do 103 k=1,n2
        j1=nn2+n2+k
        j2=n2+k
        j3=nn2+k
        j4=k
        do 101 i=1,n2
        outbuf(i1)=inbuf(j1)
        outbuf(i2)=inbuf(j2)
        j1=j1+n
        j2=j2+n
        i1=i1+1
        i2=i2+1
        outbuf(i3)=inbuf(j3)
        outbuf(i4)=inbuf(j4)
        j3=j3+n
        j4=j4+n
        i3=i3+1
        i4=i4+1
101     continue
        i1=i1+n2
        i2=i2+n2
        i3=i3+n2
        i4=i4+n2
103     continue
c
        return
        end
```

Tabelle 1. Bitreverse-Tabelle für 3 Bit

Index	Zahl	binär	binär br	Zahl br	Index br
1	0	000	000	0	1
2	1	001	100	4	2
3	2	010	010	2	3
4	3	011	110	6	4
5	4	100	001	1	5
6	5	101	101	5	6
7	6	110	011	3	7
8	7	111	111	7	8

und BILD2 überlagert, so daß niemals beide Bildtypen gleichzeitig im Speicher benutzt werden können. Beim Arbeiten mit dem einen Typ muß der andere Typ auf einer RAM-Disk gespeichert werden.

Befehle für Integer-Bilder wirken in der Regel auf die zwei Operanden BILD1 und BILD2 derart, daß ein Bild algorithmisch in das andere gewandelt wird. Für das Real-Bild wird der Datenfluß zwischen Speicher und RAM-Disk durch Ein-Operanden-Befehle gesteuert. Die Größe der Bilder (Ortsauflösung) wird durch den Parameter PICMAX und der Grauwertumfang durch PGWMAX bestimmt. Diese und weitere Parameter wie Art der Prompt-Symbole, Look-up-Tabelle, Histogramm und Hilfsvektoren werden durch das in *Bild 9* dargestellte Include-File zum Compilieren anderer Programme bereitgestellt und zur Laufzeit in der Subroutine INIT wertemäßig festgelegt. Die Bildgröße PICMAX wird wesentlich durch den verfügbaren Hauptspeicher und die Adressierungsart des Compilers bestimmt. Auf dem Zenith advanced PC 240 können mit Professional-FORTRAN 2 Integer-Bilder der Größe 256 × 256 bearbeitet werden. Das Real-Bild REBILD ist entsprechend dimensioniert.

Um ein leicht zu wartendes und leicht erweiterbares Software-System zu erstellen, das für unterschiedlichen Speicherausbau und verschiedenartige Leistungsfähigkeit von Linkern geeignet ist, wird für jeden einzelnen einfachen Befehl ein Unterprogramm erstellt. Stellt man schnelle Programme zum Transfer der Integer- und Real-Bilder zwischen RAM-Disk und Hauptspeicher zur Verfügung, so können quasi in Overlay-Technik aus unterschiedlichen vielen Teilprogrammen mehrere ausführbare Programme (MS-DOS xxx.EXE/xxx.COM) gebildet werden, die auf einen gemeinsamen Datenbestand zugreifen. Die Größe dieser Programme hängt im wesentlichen vom Speicherausbau und der Leistungsfähigkeit des Linkers ab. Im vorliegenden Fall können neben einer Bibliothek zur Bildschirmansteuerung nach ANSI-Norm ca. 20 Unterprogramme eingebunden werden.

```
C       PROGRAMM PCBILD FUER ZENITH ADVANCED PC 240
C
C       VON KLAUS-DIETER BRINKMANN
C       COMPILER: PROFESSIONAL FORTRAN
C       VERSION 3       20. FEBRUAR 1987
C
        INCLUDE 'PCBILD.COM'
C
C + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - +
C
        INTEGER*2 BILD1(PICMAX,PICMAX),BILD2(PICMAX,PICMAX)
        REAL*4 REBILD(PICMAX,PICMAX)
C
        EQUIVALENCE (BILD1,REBILD)
C
C + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - +
C
        INTEGER*4 HISUM,NEWHIS(PGWMAX),NEWSUM
        INTEGER FRAGE
        CALL INIT
C
   10 CALL CMDSTR(IJUMP,IPARA,RPARA)
      GOTO(20,40,90,100,110,120,130,140,150,170,180,200,210,220,230,240
     1,260,280,290,300,310,320,330,350,360,370,380,390,400,410,440,490,
     2510,530,550,560,570,580,590,610), IJUMP
        GO TO 10
C
C-----------------------------------------------
C
C       BEFEHL : BITGEN
C
   20 IF (IPARA.EQ.2) GO TO 30
      CALL INFO(3)
      CALL BITGEN(BILD1)
      GO TO 10
   30 CALL INFO(3)
      CALL BITGEN(BILD2)
      GO TO 10
C-----------------------------------------------
C
C       BEFEHL : COPVLT
C
   40 CALL INFO(IPARA)
      GO TO (50,60,70,80), IPARA
   50 CALL COPVLT(BILD1,BILD2)
      GO TO 10
   60 CALL COPVLT(BILD2,BILD1)
      GO TO 10
   70 CALL COPVLT(BILD1,BILD1)
      GO TO 10
   80 CALL COPVLT(BILD2,BILD2)
      GO TO 10
```

Bild 8. Ausschnitt vom Programm zur Ablaufsteuerung

```
C       INCLUDE-FILE   PCBILD.COM
C
C       VON KLAUS-DIETER BRINKMANN
C
C       COMPILER :  PROFESSIONAL FORTRAN
C
C
C       VERSION 4          18. FEBRUAR 1987
C
C * _ * _ * _ * _ * _ * _ * _ * _ * _ * _ * _ * _ * _ * _ * _ * _ * _ * _ *
C
C -->> C O M M O M  M I T  P A R A M E T E R  F E S T L E G U N G <<--
C
        INTEGER PGWMAX, PICMAX
        PARAMETER (PGWMAX=256)
        PARAMETER (PICMAX=256)
C
        INTEGER*2 VLT(PGWMAX),WINDOW(4)
        INTEGER CMDZEI,CMDSPA
        INTEGER*4 HIS(PGWMAX),WINGRO
C
        COMMON /VLTHIS/ VLT,HIS,WINDOW,WINGRO,IGWMIN,IGWMAX
C
C
        INTEGER*2 IREAD, IWRITE, IPROT, PROMP1, PROMP2
        LOGICAL*1 BEFKEN(2)
        COMMON /UNITS/ IREAD, IWRITE
        COMMON /PROMPT/ PROMP1, PROMP2, CMDZEI,CMDSPA, BEFKEN, IBEF
C
        INTEGER*2 SCRATCH(PGWMAX)
        COMMON /SCRSCR/ SCRATCH
C
        COMMON /TEMBUF/ ITSPAL, ITZEIL, ITOP
C
        LOGICAL*1 CINPUT(50)
        COMMON /STRING/ CINPUT
C
```

Bild 9. Include-File zur Bereitstellung dimensionierter Felder

```
      SUBROUTINE CMDSTR (IJUMP,IPARA,RPARA)
C
C     VON KLAUS-DIETER BRINKMANN
C
C     COMPILER: PROFESSIONAL FORTRAN
C     VERSION 3      20. FEBRUAR 1987
C
C     COMMAND STRING INTERPRETER FUER PCBILD
C     EINGABE VON BEFEHL LIEFERT SPRUNG-VARIABLE
C
      INCLUDE 'PCBILD.COM'
C
      LOGICAL*1 COMAND(2,50),PRINFO(4)
      CHARACTER*6 BEFEHL
C
      DATA COMAND/1HB,1HG,1HC,1HV,1HD,1HF,1HD,1HP,1HD,1HL,1HD,1HF,1HD,
     11HH,1HD,1HI,1HD,1HI,1HD,1HV,1HD,1HU,1HE,1HS,1HE,1HR,1HF,1HI,1HH,
     21HE,1HH,1HI,1HL,1HP,1HP,1HR,1HP,1H*,1HR,1HA,1HR,1HE,1HR,1HG,1HR,
     3HO,1HS,1HO,1HS,1HT,1HV,1HP,1HV,1HS,1HV,1HL,1HW,1HR,1HW,1HU,1HN,
     41HO,1HR,1HC,1HW,1HC,1HC,1HP,1HA,1HD,1HS,1HU,1HM,1HU,1HD,1HI,1HK,
     51HR,1HD,1HR,20*1H-/
C
      DATA PRINFO/1HI,1HR,1HC,1HK/
C
C
      IJUMP=-1
C
C       ZEILE LOESCHEN
   10 CALL CLRZEI(CMDZEI)
      CALL PLTIGW(CMDZEI,CMDSPA,PROMP1,1,0)
C
      READ (IREAD,5000,ERR=20)BEFEHL
C
      IF (BEFEHL.EQ.'BITGEN') IJUMP=1
      IF (BEFEHL.EQ.'COPVLT') IJUMP=2
      IF (BEFEHL.EQ.'DEFINE') IJUMP=3
      IF (BEFEHL.EQ.'DEPOSI') IJUMP=4
      IF (BEFEHL.EQ.'DILATA') IJUMP=5
      IF (BEFEHL.EQ.'DISFFT') IJUMP=6
                    .
                    .
      IF (BEFEHL.EQ.'xxxxxx') IJUMP=nn
C
      IF (IJUMP.GT.0) GO TO 30
C
   20 WRITE (IWRITE,5010)
      CALL INFO(-1)
      GO TO 10
C
C       VERTEILUNG NACH HILFS-PARAMETER UND BEFEHLSKENNUNG SETZEN
C
   30 BEFKEN(1)=COMAND(1,IJUMP)
      BEFKEN(2)=COMAND(2,IJUMP)
      GO TO (50,50,70,40,50,60,40,40,50,40,50,40,50,60,70,50,50,60,70,
     140,90,60,70,40,40,60,70,40,90,50,50,90,90,50,50,50,50,70,120,50),
     2  IJUMP
```

Andernfalls treten Probleme beim Linken auf (trotz 20-MB-Festplatte ist nach Linker-Fehlermeldung die Datei VM.TMP zu klein). Die Ansteuerung der einzelnen Unterprogramme erfolgt durch ein Verteilerprogramm der Art von Bild 8 in Zusammenarbeit mit dem Command-String-Interpreter von *Bild 10*. Dieser einfache Interpreter vergleicht einen eingegebenen String mit vordefinierten Befehlen und bestimmt hieraus die Sprungvariable IJUMP. In Abhängigkeit von der Befehlsart können zusätzlich Integer-, Real- oder Character-Variablen einzeln oder paarweise eingelesen werden. Über die Variablen IPARA, RPARA und CINPUT im COMMON/STRING/ werden zu rufende Unterprogramme, wie in *Bild 11* gezeigt, mit den Bildbereichsdaten und Zusatzvariablen versorgt. Wird festgestellt, daß ein nicht zum aktuellen Programm gehöriges Unterprogramm aufgerufen werden muß, werden die Bildbereiche auf die RAM-Disk kopiert und dem Benutzer wird angezeigt, welches Programm anschließend zu starten ist. Kontrolldaten auf der RAM-Disk gestatten automatisches Laden der Bildbereiche (auf automatisches „chaining" wird wegen Transportabilität zu anderen Betriebssystemen verzichtet).

In *Bild 12* wird der durchgehende Aufbau von Bildverarbeitungsprogrammen am Beispiel des in Teil 1 beschriebenen Laplace-Operators aufgezeigt. Dieses Programm kann für eigene Entwicklungen als Mustervorlage dienen, die lediglich an die eigenen Rechenvorschriften angepaßt werden muß. Der Laplace-Operator berechnet aus einem Eingangsbild jeweils mit den Nachbardaten eines Bildpunktes ein Ausgangsbild. Da für die Randpunkte keine Berechnung sinnvoll

```
C
C       KEIN PARAMETER
C
   40 IPARA=0
      RPARA=0.0
      GO TO 140
C
C       EINLESEN VON INTEGER PARAMETER
C
   50 WRITE (IWRITE,5020)PROMP2,PRINFO(1)
      READ (IREAD,5030,ERR=50)IPARA
      GO TO 140
C
C       EINLESEN VON REAL PARAMETER
C
   60 WRITE (IWRITE,5020)PROMP2,PRINFO(2)
      READ (IREAD,5040,ERR=60)RPARA
C
C       EINLESEN VON INTEGER UND REAL PARAMETER
C
   70 WRITE (IWRITE,5020)PROMP2,PRINFO(1)
      READ (IREAD,5030,ERR=70)IPARA
   80 WRITE (IWRITE,5020)PROMP2,PRINFO(2)
      READ (IREAD,5040,ERR=80)RPARA
      GO TO 140
C
C       EINLESEN VON INTEGER UND CHARACTER-STRING
C
   90 WRITE (IWRITE,5020)PROMP2,PRINFO(1)
      READ (IREAD,5030,ERR=90)IPARA
  100 WRITE (IWRITE,5020)PROMP2,PRINFO(3)
      DO 110 LX=1,50
  110 CINPUT(LX)=' '
      READ (IREAD,5050,ERR=100)(CINPUT(LX),LX=1,50)
      GO TO 140
C
C       EINLESEN VON INTEGER UND KERNEL-NUMMER
C       KERNEL-NUMMER WIRD ALS RPARA UEBERGEBEN
C
  120 WRITE (IWRITE,5020)PROMP2,PRINFO(1)
      READ (IREAD,5030,ERR=120)IPARA
  130 WRITE (IWRITE,5020)PROMP2,PRINFO(4)
      READ (IREAD,5030,ERR=130)IDUMMY
      RPARA=IDUMMY
C
  140 RETURN
C
 5000 FORMAT (A6)
 5010 FORMAT (5X,'WIE BITTE ?? ')
 5020 FORMAT (1X,2A1,': ')
 5030 FORMAT (I10)
 5040 FORMAT (F20.10)
 5050 FORMAT (50A1)
      END
```

Bild 10. Command-String-Interpreter zur Bestimmung von Sprungadressen und zur Parameter-Versorgung

ist, werden diese in DO-LOOP mit Label 10 auf 0 gesetzt. Anschließend muß jeweils von 2 bis PICMAX–1 zeilen- und spaltenweise jeder Bildpunkt untersucht werden. Dieses geschieht in den Schleifen DO 20 und DO 30. Hierbei sollte aus Zeitgründen auf die Fortran-spezifische Speicherung und Adressierung geachtet werden.
Eigene Algorithmen lassen sich leicht implementieren, wenn das Statement

OUT(MZ,MS)=
4×IN(MZ,MS)–IN(MZ–1,MS).....

durch die spezifischen Operatoren ersetzt wird. Die Berechnung von Minimum und Maximum erleichtern lediglich die Berechnung von Histogrammen und Lookup-Tabellen und gehören nicht zum eigentlichen Algorithmus. Die Aufrufe CALL TEMPO (...) stellen lediglich Zusatzinformation auf dem Terminal dar und werden im nächsten Abschnitt erläutert.

Benutzeroberfläche

PCBILD erwartet vom Benutzer keinerlei Kenntnisse in Datenverarbeitung, sondern fordert lediglich die Dateneingabe, wie sie im Manual oder (in späteren Versionen) im Daten-File eines HELP-Befehles spezifiziert ist. PCBILD wird vom Benutzer durch die String-Eingabe von Befehlen gesteuert. Diese Befehle lehnen sich an englische Bezeichnungen an. Um die Bedienung benutzerfreundlich zu gestalten und zur leichteren Einbindung von eigenen Programmen, wurde auf eine Untergliederung der Befehle durch Qualifier verzichtet. Lediglich über die Variablen IPARA, RPARA und STRING können die Befehle mit Zusatzdaten versorgt werden.
Alle Befehle sind in der dritten Zeile von unten hinter dem Befehls-Prompt * einzugeben. Benötigt ein eingegebener Befehl weitere Daten, werden in den nächsten Zeilen der Daten-Prompt > gefolgt von einer Datentyp-Kennung I, R oder C zur Eingabe von Integer-, Real- oder Character-Daten und dem : dargestellt. Die Typkennung K dient zur Auswahl einer Datenstruktur für den Befehl KERNEL. Für Zwei-Operanden-Befehle dient die Integer-Eingabe der Auswahl der Operanden mit folgender Verarbeitungsrichtung:

IPARA	Datenfluß
1	BILD1 → BILD2
2	BILD2 → BILD1
3	BILD1 → BILD1
4	BILD2 → BILD2

Zur Benutzerinformation werden unten an den rechten Bildrand Kürzel der letzten fünf bearbeiteten Befehle mit Operanden als Protokoll geschrieben.

```
      SUBROUTINE KERNEL (IN,OUT,RPARA)
C
C       VERSION 1       27. FEBRUAR 1987
C
C       VON KLAUS-DIETER BRINKMANN
C
C       FILTERUNG MIT KERN VON MAXIMAL 15,15
C
        INCLUDE 'PCBILD.COM'
C + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - +
C
      INTEGER*2 IN(PICMAX,PICMAX),OUT(PICMAX,PICMAX)
C
      INTEGER*2 KERN(15,15)
      EQUIVALENCE (KERN,SCRATCH)
      INTEGER FRAGE
C
      IKERN=RPARA
      IF (IKERN.GT.0) GO TO 60
C
C       KERN INTERAKTIV DEFINIEREN
```

Bild 11. Schnittstelle zur Übergabe von Bildbereichen und einer Zusatzvariablen RPARA

```
      SUBROUTINE LAPLAC (IN,OUT)
C
C       VERSION 2       20. FEBRUAR 1987
C
C       VON KLAUS-DIETER BRINKMANN
C
C       3*3 LAPLAC
C
        INCLUDE 'PCBILD.COM'
C + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - +
C
      INTEGER*2 IN(PICMAX,PICMAX),OUT(PICMAX,PICMAX)
C
C       RAND NULL SETZEN
C
      DO 10 M=1,PICMAX
      OUT(1,M)=0
      OUT(M,1)=0
      OUT(PICMAX,M)=0
      OUT(M,PICMAX)=0
   10 CONTINUE
C
      IGWMIN=PGWMAX
      IGWMAX=0
      CALL TEMPO(0,16)
C
C       BILD BEARBEITEN
C
      DO 30 MS=2,PICMAX-1
      IF (MOD(MS,16).EQ.0) CALL TEMPO(MS,16)
      DO 20 MZ=2,PICMAX-1
      OUT(MZ,MS)=4*IN(MZ,MS)-IN(MZ-1,MS)-IN(MZ+1,MS)-IN(MZ,MS-1)-IN(MZ,
     1S+1)
      IGWMIN=MINO(IGWMIN,OUT(MZ,MS))
      IGWMAX=MAXO(IGWMAX,OUT(MZ,MS))
   20 CONTINUE
   30 CONTINUE
C
C       EXTREMWERTE AUSGEBEN
C
      WRITE (IWRITE,5000)IGWMIN,IGWMAX
      CALL TEMPO(-10,0)
      RETURN
C
 5000 FORMAT (2X,'MINIMUM MAXIMUM ',2I5)
      END
```

Bild 12. Standardprogrammaufbau am Beispiel des Laplace-Operators

Obwohl der Zenith advanced PC240 mit eingebautem und von Professional-Fortran unterstützten mathematischen Coprozessor 80287 enorme Rechenleistung zur Verfügung stellt, dauern manche Algorithmen einige Minuten. Zur Anzeige des ordnungsgemäßen Arbeitens wird deshalb bei Beginn von langwierigen Algorithmen mit dem Unterprogramm TEMPO ( ) eine 16 Zeichen hohe Säule auf das Terminal gezeichnet, die sich allmählich verkürzt. Hierdurch kann der Benutzer einerseits beobachten, ob das Programm funktioniert (keine Verkürzung der Säule bedeutet: Rechner steht oder Endlos-Schleife) andererseits kann bei zeitintensiven Programmen das Ende abgeschätzt werden, um vielleicht zwischendurch andere Aufgaben wahrzunehmen.

Literatur

[1] *Cooley, J. W., Tukey, J. U.:* An Algorithm for the Machine Calculation of Complex Fourier Series, Math. of Comput., 4/1965.

[2] *Gonzales, R. C., Wintz, P.:* Digital Image Processing, Addison-Wesley Publishing Comp. Inc., 1979.

Erich Esders

Arbeiten mit dem 80287

Teil 2: Die Befehle

Während im ersten Teil ein Überblick über den Prozessor gegeben wurde, geht es jetzt in Details der Programmierung. Sie werden sehen, daß Ihr Computer mit einem Arithmetik-Chip sagenhaft stark wird. Die Beschreibung gilt in vielen Details auch für den 8087, der in XTs eingesteckt werden kann.

Im letzten Teil ging es zum Schluß um die Adressierungsarten bei F-Befehlen. *Bild 1* zeigt dies am Beispiel der FP-Addition 'FADD'. Die Möglichkeit einer solchen Auswahl sowie deren Optionen werden durch folgende Notation gekennzeichnet:

FADD < src | dst,src >

Der Inhalt zwischen den Spitzklammern '<' und '>' ist optional; wird er angegeben, kann zwischen einzelnen, durch senkrechte Striche voneinander getrennten Möglichkeiten gewählt werden.

Daten-Transfers

Die Instruktionen dieser Klasse transferieren Daten zwischen den internen Registern sowie dem obersten Stack-Element und dem externen Speicher. Der NPX stellt Transfer-Instruktionen für die Datentypen REAL, INTEGER und PACKED DECIMAL zur Verfügung, wie es in *Bild 2* gezeigt ist.

FLD (LoaD real)

FLD lädt den angegebenen Operanden auf den TOS, indem zunächst einmal der Stackpointer ST dekrementiert und anschließend src in das so adressierte Register geladen wird. Damit ist dieser Ladevorgang einem PUSH vergleichbar! Operand src können Stackregister und Speicher-REALs sein; der Wert letzterer wird automatisch in das temporäre REAL-Format gewandelt. Die Instruktion FLD ST(0) dupliziert beispielsweise den TOS-Wert.

FST (STore real)

FST transferiert den TOS-Wert nach dst, einem anderen Stackregister oder einer Speicher-REAL. Ist TOS ein gültiger Wert, so erfolgt vor dem Transfer eine Rundung gemäß den RC-Bits im CONTROL-Register. Zeigen jedoch die TAG-Bits für den TOS einen der Spezialfälle Unendlich, NaN oder Denormal an, schneidet der NPX stattdessen Mantisse und Exponent rechtsseitig ab, um sie für das Zielformat passend zu machen. Auf diese Weise wird der Status des Wertes für eine spätere Identifikation bewahrt!

FSTP (STore real and Pop)

FSTP wirkt in gleicher Weise wie FST, nur das nach dem Transfer der aktuelle TOS-Wert einen Tag-Wert „Leer" erhält und anschließend der Stackpointer ST inkrementiert wird. Beispielsweise bewirkt die Instruktion FSTP ST(0) einen POP ohne Datentransfer!

FXCH (eXCHange registers)

FXCH ohne Operandenangabe tauscht die Inhalte der beiden oberen Stackregister aus. Mit Hilfe dieser Instruktion kann man einfach beliebige Stackregister Operationen unterziehen, ohne den TOS-Wert zu modifizieren. Um beispielsweise den Inhalt des viertobersten Stackregisters durch seinen Quadratwurzel-Wert zu ersetzen, kodiert man:

```
FXCH ST(4)
FSQRT
FXCH ST(4)
```

FILD (Integer LoaD)

FILD wandelt eine Speicher-INTEGER in das temporäre REAL-Format und lädt diesen Wert in das TOS-Register. Das zugehörige Tag indiziert 0, wenn alle Bits der src eine „0" enthielten, im anderen Fall ein Gültig.

FIST (Integer STore)

FIST rundet den TOS-Wert gemäß RC-Feld in eine INTEGER und transferiert das Resultat in eine Speicher-INTEGER. Sowohl die negative als auch die positive Null zeigen das gleiche Format: 000...00!

FISTP (Integer STore and Pop)

FISTP wirkt in gleicher Weise wie FIST, nur

```
- implizit      FADD           addiert die oberen beiden
                               Stack-Elemente und weist
                               das Ergebnis dem obersten zu.

- implizit/     FADD src       addiert das oberste Stack-
  explizit                     Element und 'src' und weist
                               das Ergebnis dem TOS zu.

- explizit      FADD dst,src   addiert 'dst' und 'src' und
                               weist das Ergebnis 'dst' zu.
```

Bild 1. Operanden können implizit angegeben werden, explizit oder in manchen Fällen wahlweise implizit oder explizit

```
FLD   src       Lädt REAL-Operand
FST   dst       Speichert REAL-Operand
FSTP  dst       Speichert REAL-Operand und POP
FXCH  <dst>     Tauscht Stackregister-Inhalte

FILD  src       Lädt INTEGER-Operand
FIST  dst       Speichert INTEGER-Operand
FISTP dst       Speichert INTEGER-Operand und POP

FBLD  src       Lädt PACKED DECIMAL-Operand
FBSTP dst       Speichert PACKED DECIMAL-Operand und POP
```

Bild 2. Das sind die Datentransfer-Befehle

das nach dem Transfer der aktuelle TOS-Wert einen Tag-Wert „Leer" erhält und anschließend der Stackpointer ST inkrementiert wird.

FBLD (packed decimal Bcd LoaD)

FBLD wandelt den Wert einer Speicher-PACKED DECIMAL in das temporäre REAL-Format und weist das Ergebnis dem TOS-Register zu. Sowohl Vorzeichen als auch der Fall -0 bleiben bei der exakten Konvertierung ohne Rundungsfehler erhalten. Diese Instruktion nimmt keine Überprüfung der einzelnen Nibbles auf Pseudotetraden vor!

FBSTP (packed decimal Bcd STore and Pop)

FBSTP konvertiert den TOS-Wert in das PACKED DECIMAL-Format und transferiert das Resultat in eine gleichnamige Speichervariable; danach wird ST inkrementiert. Diese Instruktion erzeugt einen gerundeten Wert, indem nach Addition einer 0.5 das Resultat abgeschnitten wird!

Arithmetik

Die Tabelle aus *Bild 3* enthält eine Übersicht aller arithmetischen Instruktionen. Für die vier Grundrechenarten sowie die implementierten Funktionen kann der NPX als klassische Stack-Maschine arbeiten; in diesem Fall besitzt die Instruktion keine explizite Angabe von Operanden. Das TOS-Register und evtl. sein Nachfolger dienen als Quelloperanden, Ergebnisse werden immer dem TOS-Register wieder zugewiesen. Wenn benötigt, kann jedoch auch eine Register-orientierte Arbeitsweise programmiert werden, indem entweder das TOS- und ein weiteres Register, oder aber zwei explizit angegebene Register des Stacks als Operanden angegeben werden.

FADD (ADD real)
FADDP (ADD real and Pop)
FIADD (Integer ADD)

Diese Instruktionen addieren Quell- und Zieloperanden und weisen das Ergebnis dem Zieloperanden zu. Mit Hilfe der Instruktion FADD ST,ST(0) wird beispielsweise der TOS-Wert verdoppelt!

FSUB (SUBtract real)
FSUBP (SUBtract real and Pop)
FISUB (Integer SUBtract)

Die „normalen" Subtraktions-Instruktionen vermindern den Wert des Ziel- durch denjenigen des Quell-Operanden:

```
FADD    < src ¦ dst,src >   Addiert REAL
FADDP     dst ¦ src         Addiert REAL und POP
FIADD     src               Addiert INTEGER

FSUB    < src ¦ dst,src >   Subtrahiert REAL
FSUBP     dst ¦ src         Subtrahiert REAL und POP
FISUB     src               Subtrahiert INTEGER

FSUBR   < src ¦ dst,src >   Subtrahiert umgekehrt REAL
FSUBRP    dst ¦ src         Subtrahiert umgekehrt REAL und POP
FISUBR    src               Subtrahiert umgekehrt INTEGER

FMUL    < src ¦ dst,src >   Multipliziert REAL
FMULP     dst,src           Multipliziert REAL und POP
FIMUL     src               Multipliziert INTEGER

FDIV    < src ¦ dst,src >   Dividiert REAL
FDIVP     dst,src           Dividiert REAL und POP
FIDIV     src               Dividiert INTEGER

FDIVR   < src ¦ dst,src >   Dividiert umgekehrt REAL
FDIVRP    dst,src           Dividiert umgekehrt REAL und POP
FIDIVR    src               Dividiert umgekehrt INTEGER

FSQRT                       Quadratwurzel-Berechnung
FSCALE                      Multiplikation mit Zweier-Potenzen
FPREM                       Modulo-Funktion
FRNDINT                     INTEGER-Rundung

FXTRACT                     REAL-Zerlegung
FABS                        Absolutwert-Bildung
FCHS                        Vorzeichen-Negierung
```

Bild 3. Die arithmetischen Instruktionen

dst := dst – src

FSUBR (SUBtract real Reverse)
FSUBRP (SUBtract real Reverse and Pop)
FISUBR (Integer SUBtract Reverse)

Die „umgekehrten" Subtraktions-Instruktionen vermindern den Wert des Quell- durch denjenigen des Ziel-Operanden und weisen das Ergebnis dem letzteren zu:

dst := src – dst

FMUL (MULtiply real)
FMULP (MULtiply real and Pop)
FIMUL (Integer MULtiply)

Diese Instruktionen multiplizieren Quell- und Zieloperanden und weisen das Ergebnis dem Zieloperanden zu. Mit Hilfe der Instruktion FMUL ST,ST(0) wird beispielsweise der TOS-Wert quadriert!

FDIV (DIVide real)
FDIVP (DIVide real and Pop)
FIDIV (Integer DIVide)

Die „normalen" Divisions-Instruktionen dividieren den Wert des Ziel- durch denjenigen des Quell-Operanden und weisen das Ergebnis ersterem zu:

dst := dst / src

FDIVR (DIVide real Reverse)
FDIVRP (DIVide real Reverse and Pop)
FIDIVR (Integer DIVide Reverse)

Die „umgekehrten" Divisions-Instruktionen dividieren den Wert des Quell- durch denjenigen des Ziel-Operanden und weisen das Ergebnis dem letzteren zu:

dst := src / dst

FSQRT (SQuare RooT)

FSQRT ersetzt den Inhalt des TOS-Registers durch dessen Quadratwurzel-Wert.

FSCALE (SCALE)

Diese Instruktion interpretiert den Inhalt des Stackregisters ST(1) als INTEGER und addiert dessen Wert zum Exponenten des TOS- Registers:

$ST(0) := ST(0) * 2^{ST(1)}$

Hierdurch erhält man die Möglichkeit einer sehr schnellen Multiplikation mit oder Division durch ganzzahlige Potenzen von 2! FSCALE nimmt an, daß der Inhalt aus ST(1) eine ganze Zahl im Bereich $-2^{15} \leq ST(1) \leq 2^{15}$ ist. Handelt es sich aber um eine reelle Zahl dieses Bereiches, deren Betrag größer 1 ist, rundet FSCALE vor der Addition auf die nächst niedrigere INTEGER. In allen anderen Fällen liefert die Instruktion undefinierte Resultate. Sicherheitshalber lädt man ST(1) vor der Ausführung von FSCALE mit einer Speicher-INTEGER und umgeht so alle Probleme!

FPREM (Partial REMainder)

FPREM führt eine Modulo-Division des

Condition Code				Interpretation after FPREM
C3	C2	C1	C0	
X	1	X	X	Incomplete Reduction; further iteration is required for complete reduction.
				Complete Reduction; C1, C3, and C0 contain the three least-significant bits of quotient:
0	0	0	0	(Quotient) MOD 8 = 0
0	0	0	1	(Quotient) MOD 8 = 4
0	0	1	0	(Quotient) MOD 8 = 1
0	0	1	1	(Quotient) MOD 8 = 5
1	0	0	0	(Quotient) MOD 8 = 2
1	0	0	1	(Quotient) MOD 8 = 6
1	0	1	0	(Quotient) MOD 8 = 3
1	0	1	1	(Quotient) MOD 8 = 7

Bild 4. Nach FPREM müssen die Bit C0...C3 so interpretiert werden

```
FCOM      < src >     Vergleicht REAL
FCOMP     < src >     Vergleicht REAL und POP
FCOMPP                Vergleicht REAL und POP zweimal
FICOM       src       Vergleicht INTEGER
FICOMP      src       Vergleicht INTEGER und POP
FTST                  Testet TOS-Wert
FXAM                  Untersucht TOS-Wert
```

Bild 5. Die Vergleichs-Befehle

Condition Code				Interpretation after FCOM
C3	C2	C1	C0	
0	0	X	0	ST > source
0	0	X	1	ST < source
1	0	X	0	ST = source
1	1	X	1	ST is not comparable

Bild 6. Nach FCOM wird Condition Code wie folgt gesetzt

TOS-Wertes mit dem Inhalt des nächsten Stackregisters durch; ST(1) ist also der Modulus. Das dabei entstehende Resultat ist exakt, da die Instruktion den Rest durch fortlaufende Subtraktion berechnet. Bei extrem großen Unterschieden in den Beträgen der beiden Operanden, kann dieser Vorgang eine recht lange Zeit in Anspruch nehmen. Da der NPX nur zwischen zwei Instruktionen angesprochen werden kann, erhöht sich in solchen Fällen die Interrupt-Latenzzeit extrem! Ursprünglich wurde diese Instruktion daher für die Verwendung in einer Software-gesteuerten Programmschleife entworfen. FPREM verkleinert die Differenz beider Operanden-Beträge während einer Ausführung um 2^{64}. Entsteht dabei ein Rest kleiner als der Modulus, ist die Funktion erfolgreich beendet und C2 im CONDITION CODE wird auf „0" gesetzt. Im anderen Fall enthält C2 eine „1" und das Ergebnis wird partieller Rest genannt. Die übergeordnete Software-Schleife kann nun eine weitere Ausführung veranlassen, bis der entgültige Rest berechnet ist.

Ein weiteres Kriterium für die Schleifen-Überwachung wäre auch ein Operanden-Vergleich der beiden obersten Stackregister. Weiterhin stellt FPREM mit dem endgültigen Rest dessen drei niederwertigste Bits in C0, C1 und C3 zur Verfügung (Bild 4).

FRNDINT (RouND to INTeger)

Gemäß dem RC-Feld im CONTROL-Register rundet FRNDINT den TOS-Wert zur nächst höheren bzw. niedrigeren Ganzzahl.

FXTRACT (eXTRACT exponent and significand)

Diese Instruktion „zerlegt" den Inhalt des TOS-Registers dergestalt, daß Exponent und Mantisse getrennt in zwei REAL-Zahlen gewandelt und ersterer im TOS- und die Mantisse im nächst folgenden Stackregister abgelegt werden. Die Resultate repräsentieren die wahren, d.h., ohne die Offsets behafteten Werte des ursprünglichen TOS-Inhaltes! Sehr nützlich kann FXTRACT in Verbindung mit FBSTP angewandt werden, wenn REAL-Zahlen in ihre dezimale Darstellung zu Ausgabezwecken umgewandelt werden müssen. Eine weitere Anwendung liegt im Debugging, können doch so Exponent- und Mantissen-Teil beliebiger Resultate getrennt untersucht werden.

FABS (ABSolute value)

FABS ersetzt den TOS-Wert durch seinen Absolut-Wert, indem diese Instruktion dessen Vorzeichen auf positiv setzt.

FCHS (CHange Sign)

FCHS negiert das Vorzeichen des TOS-Wertes.

Vergleiche

Die Instruktionen aus *Bild 5* testen das TOS-Register oder vergleichen es mit einem anderen Operanden und legen das Ergebnis im CONDITION CODE des STATUS REGISTERs ab. Mit Hilfe der weiter unten besprochenen Instruktion FSTSW kann der Inhalt dieses Register in den externen Speicher transferiert und so nach einem Vergleich analysiert werden.

FCOM (COMpare)

FCOM vergleicht den Wert des TOS-Registers mit einem zweiten Operanden. Wenn nicht explizit in Form eines Registers oder einer Speicher-Variablen angegeben, wird ST(1) angenommen. Die Instruktion behandelt +0 und -0 gleich und setzt den CONDITION CODE gemäß der Tabelle aus *Bild 6.*

FCOMP (COMpare and Pop)

FCOMP ist mit FCOM identisch, zusätzlich wird ST inkrementiert.

FCOMPP (COMpare and PoP twice)

FCOMPP ist mit FCOM ohne Parameter-Angabe identisch. Die Instruktion vergleicht die beiden obersten Stack-Werte miteinander und inkrementiert anschließend den Stackpointer ST zweimal.

FICOM (Integer COMpare)

FICOM wandelt src in das temporäre REAL-Format und vergleicht das Resultat mit dem Inhalt des TOS-Registers.

FICOMP (Integer COMpare and Pop)

FICOM wandelt src in das temporäre REAL-Format, vergleicht das Resultat mit dem Inhalt des TOS-Registers und inkrementiert anschließend den Stackpointer.

FTST (TeST)

FTST ist mit FCOM ohne Parameter-Angabe identisch und vergleicht den Inhalt des TOS-Registers mit 0! Das Resultat entspricht der Tabelle aus Bild 21 mit source = 0.

FXAM (eXAMine)

Diese Instruktion untersucht den Inhalt des TOS-Registers auf Vorzeichen, NaN, 0, Normale oder Leer. Die Tabelle aus Bild 7 zeigt den resultierenden CONDITION CODE. Auffällig ist die redundante Kodierung für „LEER"; für eine Abfrage auf diesen Fall sollten nur die Bits C1 und C2 herangezogen werden.

Condition Code				Interpretation
C3	C2	C1	C0	
0	0	0	0	+ Unnormal
0	0	0	1	+ NaN
0	0	1	0	− Unnormal
0	0	1	1	− NaN
0	1	0	0	+ Normal
0	1	0	1	+ ∞
0	1	1	0	− Normal
0	1	1	1	− ∞
1	0	0	0	+ 0
1	0	0	1	Empty
1	0	1	0	− 0
1	0	1	1	Empty
1	1	0	0	+ Denormal
1	1	0	1	Empty
1	1	1	0	− Denormal
1	1	1	1	Empty

Bild 7. Condition Code nach FXAM

Transzendente

Die Instruktionen dieser Klasse (*Bild 8*) führen die sonst zeitintensiven Berechnungen trigonometrischer Funktionen durch. Sie arbeiten mit den beiden obersten Stack-

FPTAN	*partieller Tangens*
FPATAN	*partieller Arcustangens*
F2XM1	$2^X - 1$
FYL2X	$Y * log_2(X)$
FYL2XP1	$Y * log_2(X+1)$

Bild 8. Die transzendenten Funktionen

registern und liefern das Resultat jeweils im TOS ab. Lediglich zur Bereichsjustierung der Argumente bzw. Resultate ist ein zusätzlicher Software-Aufwand nötig, da alle Instruktionen annehmen, ihre Operanden seien gültig und im korrekten Zahlenbereich!

FPTAN (Partial TANgent)

Diese Instruktion berechnet die Funktion Y/X = TAN(α). Das Argument alpha ist der Inhalt des TOS-Registers und muß Element des Bereiches $0 \leq \alpha \leq \pi/4$ sein. Das Resultat der Operation ist ein Bruch, dessen Zähler Y den aktuellen TOS-Wert alpha überschreibt. Der Nenner X wird durch einen PUSH neues TOS-Element. In Verbindung mit dem gebrochenen Argument der inversen Funktion Arcustangens gestattet diese Art der Berechnung eine vereinfachte Berechnung weiterer trigonometrischer Funktionen, wie es *Bild 9* zeigt.

FPATAN (Partial ArcTANgent)

FPATAN berechnet die Funktion α = ARCTAN(Y/X), wobei X aus dem TOS-Register und Y aus seinem Nachfolger entnommen wird. Für X und Y muß gelten: $0 \leq Y < X < \infty$. Die Instruktion führt auf dem Stack einen POP durch und legt das Resultat α im TOS ab, wodurch Y überschrieben wird!

$$\tan \alpha = \frac{\sin \alpha}{\cos \alpha} = \frac{1}{\cot \alpha}$$

$$\arctan \frac{\alpha}{\sqrt{1 - \alpha^2}} = \arcsin \alpha = \frac{\pi}{2} - \arccos \alpha$$

Bild 9. Ableitungsbeispiele für tan und arctan

F2XM1 (2 to the power of X Minus 1)

F2XM1 berechnet die Funktion $Y = 2^{(X-1)}$. X ist dabei der Inhalt des TOS-Registers und muß im Bereich $0 \leq X \leq 0.5$ liegen; das Resultat Y wird im TOS abgelegt. Diese Instruktion berechnet auch für sehr kleine X ein sehr genaues Y! Addiert man eine 1.0 zum Ergebnis, entspricht dies der Berechnung $Y = 2^X$. Mit Hilfe der weiter unten beschriebenen NPX-internen Konstanten, lassen sich dann auch die in *Bild 10* gezeigten Potenzen einfach berechnen.

$$10^X = 2^{X * LOG_2(10)}$$

$$e^X = 2^{X * LOG_2(e)}$$

$$Y^X = 2^{X * LOG_2(Y)}$$

Bild 10. Potenzen, in Zweierpotenzen umgerechnet

FYL2X (Y times Log base 2 of X)

FYL2X berechnet die Funktion Z = Y * log2(X). Für die Operanden gilt:
$X : 0 < ST(0) < \infty$
$Y : \infty < ST(1) < \infty$

Die Instruktion führt einen Pop auf dem Stack durch und Z wird im TOS-Register abgelegt. Aufgrund dieser Rechenvorschrift lassen sich die Berechnungen von Logarithmen beliebiger Basen vereinfachen:

logb(X) = M * log2(X)
mit dem Umrechnungsfaktor (Modul)

M = 1/log2(b).

FYL2XP1 (Y times Log base 2 of (X+1))

Diese Instruktion berechnet die Funktion Z = Y * log2(X+1). Für die Operanden gilt:

$X : 0 \leq |ST(0)| < (1 - \sqrt{2} / 2)$
$Y : \infty < ST(1) < \infty$

Die Instruktion führt einen Pop auf dem Stack durch und Z wird im TOS-Register abgelegt. Gegenüber FYL2X bietet diese Instruktion eine höhere Genauigkeit, wenn der Logarithmus einer Zahl sehr nahe bei 1 berechnet werden soll!

Konstante

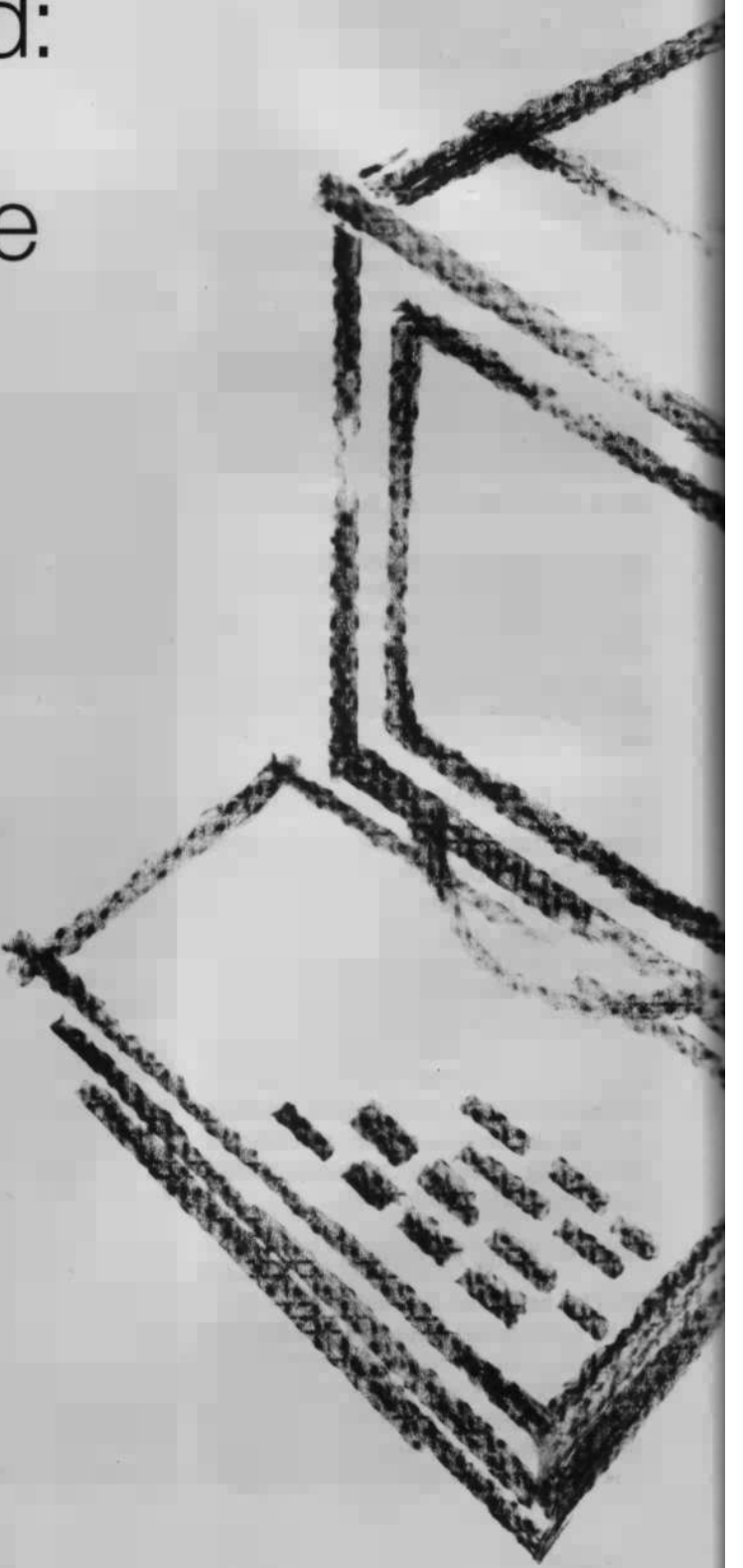
Jede der Instruktionen aus *Bild 11* lädt eine häufig benutzte Konstante in das TOS-Register. Diese vordefinierten Werte liegen im temporären Floating Point Format vor und besitzen eine Genauigkeit von 19 Dezimalstellen. Da die Instruktionen nur zwei Byte lang sind, durch ihre Ausführung aber eine 10 Byte umfassende temporäre REAL geladen werden kann, ergibt sich hier eine Ersparnis sowohl im Code-Umfang als auch in dessen Ausführungszeit!

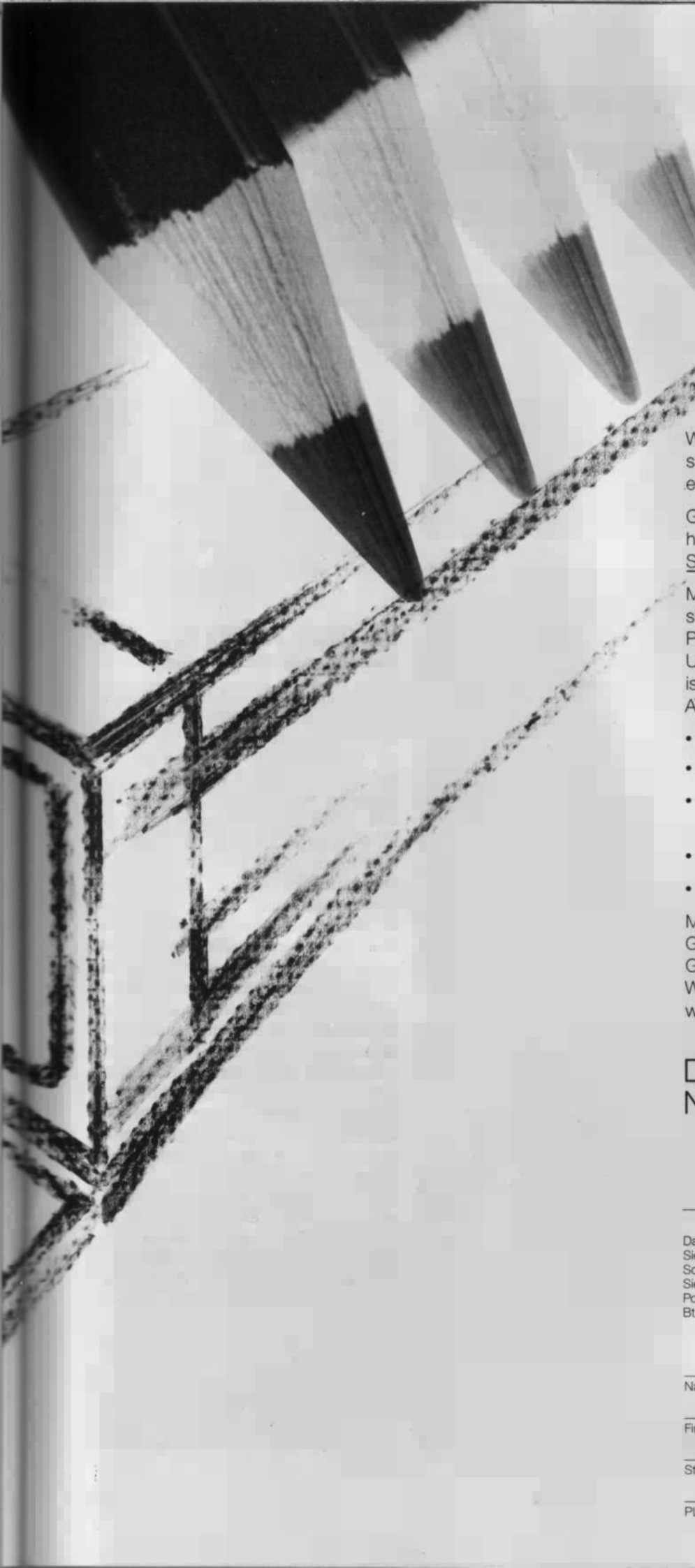
Prozessor Steuerungs-Instruktionen

Die im *Bild 12* gezeigten Instruktionen werden normalerweise nicht während regulärer Berechnungen eingesetzt. Sie dienen zur Überwachung des NPX auf System-Ebene, indem sie Initialisierungen, Exception-Behandlungen und Task-Umschaltungen unterstützen und sollen daher nur im

SIEMENS

Wenn 16 Bit zu wenig sind: Der neue, superschnelle IndustriePC





Viele Aufgaben in der Industrie erfordern extrem hohe Verarbeitungsgeschwindigkeiten – und viel Speicherplatz.

Wer schon einmal vor dem Monitor saß und auf die Daten wartete, weiß ein Lied davon zu singen ...

Genau für solche Anwendungsfälle haben wir nun den neuen IndustriePC SICOMP® PC 32-05.

Mit 32 Bit ist er so schnell, wie Sie es sich heute wünschen; sein vernünftiger Preis wird Sie überraschen.
Und weil er ein richtiger IndustriePC ist, bietet er Ihnen außer der vollen AT-Kompatibilität noch

- Schutzart nach Maß
- Realtime-Betriebssystem
- Kommunikationsfähigkeit im Automatisierungsverbund
- extrem hohe Funktionssicherheit und
- aufgabengenaue Projektierbarkeit.

Mehr Informationen?
Gern!
Greifen Sie zum Stift.
Wir senden Ihnen postwendend das Info-Paket.

progress in automation: Siemens

Die IndustriePC Natürlich von Siemens

Das kostenlose Info-Paket über die IndustriePC liegt für Sie bereit.
Schreiben Sie an
Siemens AG, Infoservice 215/Z 317 k
Postfach 2348, 8510 Fürth.
Btx: * 3200021 2614 #

Name

Firma/Abteilung

Straße/Tel.

PLZ/Ort

A 19100 - F 851 - Z 836

```
FLDZ      Lädt +0.0 in das TOS-Register
FLD1      Lädt +1.0 in das TOS-Register
FLDPI     Lädt π in das TOS-Register
FLDL2T    Lädt log2(10) in das TOS-Register
FLDL2E    Lädt log2(e) in das TOS-Register
FLDLG2    Lädt log10(2) in das TOS-Register
FLDLN2    Lädt loge(2) in das TOS-Register
```

Bild 11. Befehle zur Erzeugung von Konstanten

```
FINIT / FNINIT                Initialisiert Numerik-Prozessor
FSETPM                        Initialisiert "PROTECTED MODE"
FLDCW                 src     Lädt CONTROL-REGISTER
FSTCW / FNSTCW        dst     Speichert CONTROL REGISTER
FSTSW / FNSTSW        dst     Speichert STATUS REGISTER
FSTSW AX  /  FNSTSW AX        Speichert STATUS REGISTER nach AX
FCLEX / FNCLEX                Löscht alle Exception-Flags
FSTENV / FNSTENV      dst     Speichert Teil-NPX-Umgebung
FLDENV                src     Lädt Teil-NPX-Umgebung
FSAVE / FNSAVE        dst     Speichert komplette NPX-Umgebung
FRSTOR                src     Lädt komplette NPX-Umgebung
FINCSTP                       Inkrementiert Stackpointer
FDECSTP                       Dekrementiert Stackpointer
FFREE                 dst     Indiziert Stackregister als "LEER"
FNOP                          Keine Operation
FWAIT                         Synchronisiert NPX, CPU WAIT
```

Bild 12. Die Prozessor-Steuerbefehle

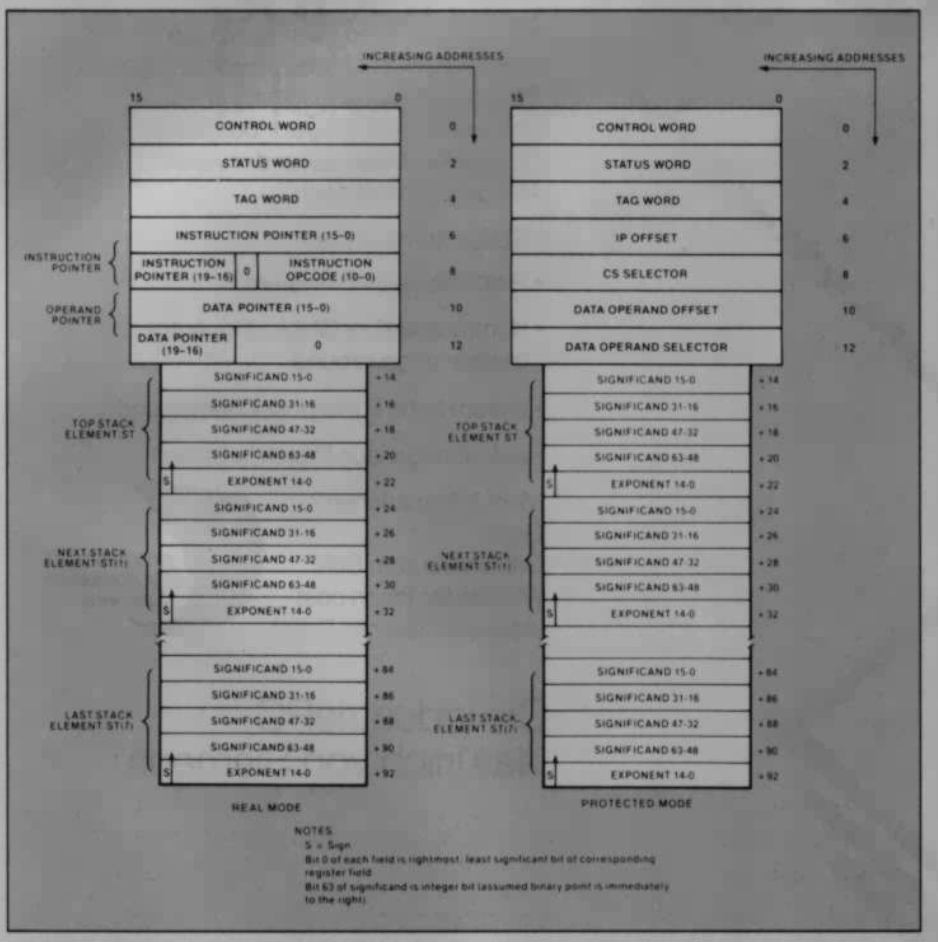


Bild 13. Mit FSAVE und FRSTOR kann der gesamte NPX-Inhalt auf den Speicher gegeben und wieder geholt werden

groben Überblick behandelt werden. Einige Steuerungs-Instruktionen unterscheiden die sogenannte „wait"- und „no-wait"-Form, erstere (Fxxxx) stellt der Instruktion einen Test der $\overline{\text{ERROR}}$-Leitung des NPX vorweg, letztere (FNxxxx) nicht! Im Gegensatz dazu testen die numerischen Instruktionen vor ihrer Ausführung stets automatisch die $\overline{\text{BUSY}}$-Leitung und erreichen so die Synchronität mit der CPU. Mit Hilfe der Instruktionen FSTCW und FSTSW läßt sich der Inhalt des Control- und Status-Registers in den externen Speicher kopieren. FSTENV transferiert den Inhalt aller Flag- sowie Pointer-Register und FSAVE die komplette NPX-Umgebung in das externe RAM. *Bild 13* zeigt eine Speicherabbildung nach einem FSAVE, wobei die Bedeutung einiger Worte im oberen Teil je nach REAL- und PROTECTED-Mode unterschiedlich ist. Diese beiden Modi entsprechen den gleichnamigen Adressierungsarten der CPU 80286. Im ersten Fall wird die vom 8086 her bekannte Weise emuliert und im letztgenannten eine virtuelle Adressierung gestattet. Ein FSTENV liefert nur diesen Teil ohne die Inhalte der einzelnen Stack-Register! Die Lade-Instruktion FLDENV gestattet die Vorbesetzung der NPX-Umgebung aus dem externen Speicher. Der Vollständigkeit halber enthält der Beispielteil zum Schluß dieses Artikels eine kurze Anwendung dieser Instruktionen. Ohne irgendein Flag oder Register zu modifizieren, kann der Stackpointer ST mit Hilfe von FINCSTP inkrementiert oder durch FDECSTP dekrementiert werden. Dabei folgt einem ST=7 ein ST=0 bzw. einem ST=0 ein ST=7!

Der Einsatz des NPX in Anwendungs-Programmen

Für den Einsatz des NPX in Anwenderprogrammen soll hier die MS-DOS-Umgebung auf einem AT vorausgesetzt werden. Für die Entwicklung der folgenden Beispiele standen die Microsoft-Werkzeuge Macro-Assembler MASM 4.00, C-Compiler MSC 4.00, Linker LINK 3.51 sowie der CODE-VIEW-Debugger CV 1.00 zur Verfügung. Mechanisch ist der 80287 schnell installiert. Nach dem obligaten „Entfernen der Gehäuse-Abdeckung" ist der freie Sockel für den Numerik-Prozessor entweder auf dem Motherboard des Rechners oder aber der CPU-Platine (in einem rein busorientierten System) lokalisiert. Das vorsichtige und feierliche Einsockeln des Chips sowie das anschließende Zusammenbauen des Systems sind dann nur noch ein „Kinderspiel" – wenn man entsprechenden Anleitungen glauben schenkt! Im nächsten Schritt ist auf einigen Rechnertypen dieser nachträgliche Einbau dem System per Konfigurierung bekannt zu machen, ansonsten kommt es bei jedem Boot-Vorgang zu einem „Configuration Error" mit der Aufforderung „Run Setup". Ab diesem Zeitpunkt kann ein Programm den NPX in seine Aktivitäten einbeziehen. Komfortable Anwendungen, die eine große Rechenleistung im Floating Point Format erbringen müssen, testen bei ihrem Start zunächst einmal die Rechner-Umgebung. Stellen sie dabei das Vorhandensein eines Numerik-Prozessors fest, führen sie für die Laufdauer des Programms alle Berechnungen mit den entsprechenden NPX-Instruktionen durch. Im anderen Fall werden diese durch passende Subroutinen

emuliert, d. h., durch Programm-Moduln „nachgeahmt"! Um diese Transparenz zu erreichen, sind selbstverständlich die Emulierung und die Original-Instruktionen völlig identisch. Das Einsetzen des NPX erkennt der Anwender dann aber durch die ungleich höhere Abarbeitungs-Geschwindigkeit seines Programmes. Weniger aufwendige Programme führen den oben beschriebenen Test nicht durch und bilden in jedem Fall die Floating Point-Instruktionen eines NPX durch Software nach. Der Nachteil ist offensichtlich: Der eingesetzte Numerik-Prozessor überwacht den Instruktions-Strom der CPU, findet keine der Extensions für sich und tritt damit auch nicht in Aktion! Entwickelt man nun selbst Numerik-Programme in MICROSOFT C für den Ablauf unter MS-DOS, enthalten die ausführbaren (name.EXE) Dateien automatisch den Test auf Vorhandensein eines NPX, sofern dieser nicht optional unterdrückt wird. Letzteres kann bei Programmen, die einen 80287 voraussetzen, gewünscht sein, da durch den Wegfall der Software-Emulierung ein umfangreiches Code-Modul in der name.EXE-Datei eingespart wird. In solchen Anwendungen sollte jedoch das Programm dennoch das Vorhandensein eines NPX prüfen und im negativen Fall mit einer entsprechenden Meldung auf der Konsole abbrechen. Für den seltenen Fall, einen Numerik-Prozessor trotz Vorhandensein nicht nutzen zu wollen, ist die Variable „NO87" in der MS-DOS-Umgebung (Environment) einzutragen. Gibt man auf System-Ebene

SET NO87=message

ein, trägt MS-DOS eine Variable mit dem Namen NO87 und dem Inhalt 'message' in das Environment ein. Ein durch MSC kompiliertes Programm durchsucht nun das Environment und gibt bei Vorhandensein der NO87-Variablen den String message aus, der auch aus nur einem Leerzeichen (SPACE) bestehen kann. Gleichzeitig werden dann alle Flaoting Point-Operationen via Software nachgebildet! Durch Eingabe von

SET NO87=

ohne jegliches Zeichen für message kann man die Variable wieder aus der MS-DOS-Umgebung austragen. Alle nachfolgenden Programme sind in ihrer reinen Textform abgedruckt und enthalten neben den nötigen Kommentaren die entsprechenden Generierungs-Informationen im Programmkopf. Letztere werden durch die Eingabezeile zum Aufruf von Assembler, Compiler oder Linker mit allen Optionen gekennzeichnet. Bei allen Vorgängen ist die nachfolgende Bearbeitung des entstehenden EXE-Files mit dem CODEVIEW-Debugger

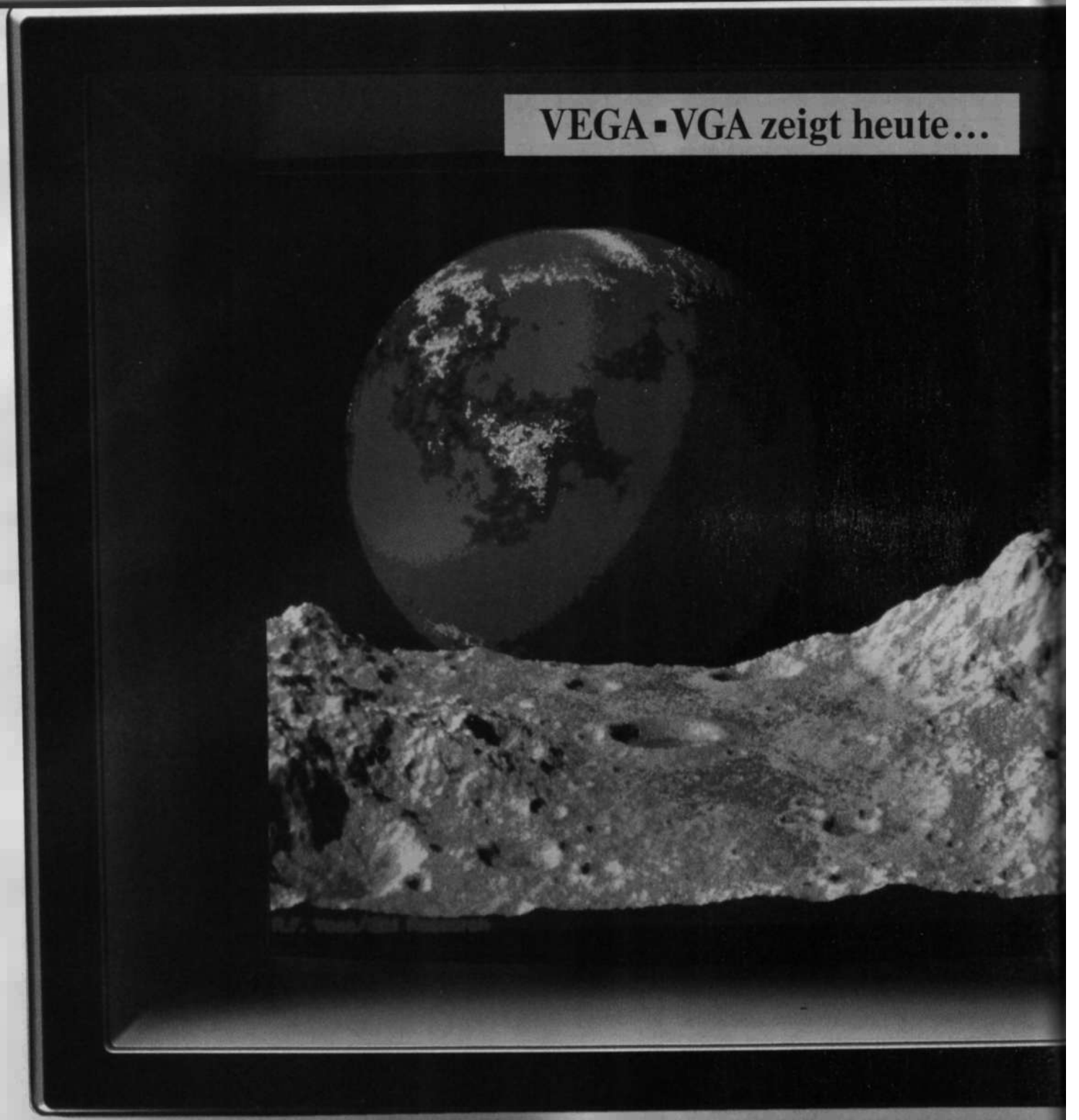
```
EXE-File für symbolisches Debugging generieren

MASM name.ASM /R;          Floating-Point-Instruktionen zulassen;
                           interne Symbole PUBLIC deklarieren;

MSC name.C /G2 /Zi /Od;    80286-Instruktionen benutzen;
                           CODEVIEW-Informationen hinzufügen;
                           keine Code-Optimierung durchführen;

LINK name.OBJ /CO;         EXE-File mit CODEVIEW-Infos generieren;


Kompakten EXE-File generieren

MASM name.ASM /R;          Floating-Point-Instruktionen zulassen;

MSC name.C /G2 /Ox;        80286-Instruktionen benutzen;
                           OBJ-File maximal optimieren;

LINK name.OBJ /EXEPACK;    kompakten EXE-File generieren;
```

Bild 14. Unterschiedliche Compiler-/Linker-Optionen

```
/*****************************************************
*   FP_EXTIM.C, Bestimmung von Floating-Point        *
*               Ausführungszeiten verschiedener      *
*               Operationen zu Vergleichszwecken!    *
*                                                    *
*   Compile  :  MSC FP_EXTIM /G2 /Zi /Od;            *
*   Link     :  LINK FP_EXTIM /CO;                   *
*   Run      :  FP_EXTIM                             *
*                                                    *
 *****************************************************/


#include        <stdio.h>
#include        <sys\types.h>
#include        <sys\timeb.h>
#include        <time.h>
#include        <math.h>

#define         IMAX    50000           /*  Anzahl Operationen          */
#define         CONST   3.33333333E33   /*  REAL-Konstante              */

struct timeb    start,
                stop;                   /*  Start-/Stop-Zeit-Strukturen  */


/*****************************************
*   test_add()   Floatingpoint-Addition!   *
 *****************************************/

test_add()

{ unsigned      i;
  double        x,
                y;

  ftime( &start );

  for( i = 0; i < IMAX; i++ )
     y = x + CONST;

  ftime( &stop );

}

/*************************************************
*   test_mul()   Floatingpoint-Multiplikation!   *
 *************************************************/

test_mul()

{ unsigned      i;
  double        x,
                y;

  ftime( &start );

  for( i = 0; i < IMAX; i++ )
     y = x * CONST;

  ftime( &stop );

}
```

Bild 15. Ein Laufzeittest in C: FP_EXTIM.C

VEGA▪VGA zeigt heute...



...256 Farben, aus einer brillanten Palette von 262144

...64 Graustufen, Videoqualität auf PS/2 Monochrom-Monitor

unretuschierte Original-Bildschirmfotos. Copyrights by Z-Soft und IBM.

...**800 x 600 Punkte,** gestochen scharf, z. B. Windows/386 auf JVC AutoScan

Garantiert!

VEGA ▪ VGA

Jetzt gibt es eine Farbkarte, mit der Sie die fantastischen Graphikmöglichkeiten der neuen IBM PS/2-Systeme auch auf Ihrem PC/XT/AT nutzen können: VEGA-VGA von Video 7 ist als erste Farbkarte kompatibel mit allen VGA-Programmen. 100 Prozent. Garantiert.

VGA, CGA, MDA, EGA, Hercules.

Sie wollen auf Ihre bewährten EGA- und Hercules-Programme nicht verzichten? No Problem. Ihren bisherigen Monitor möchten Sie weiter benutzen? OK. Sie hätten bei Symphony lieber 132 x 43 Zeichen? Auch OK. Sie brauchen bei AutoCAD mehr Durchblick? 800 x 600 VEGA-Auflösung bringt's. Sie denken schon jetzt an OS/2? Oder an Windows/386? Wir auch. Den Treiber gibt's deshalb kostenlos dazu.

Ausgezeichnet.

Mit über 500.000 EGA-Karten ist Video 7 heute der Marktführer. Video 7 wurde von führenden amerikanischen Fachzeitschriften mehrfach zur „Besten Graphik-Karte" gewählt (PC-Magazine, PC-World, PC Tech-Journal).

Wann schauen Sie sich die neue VEGA-VGA an?

- 100% VGA, EGA, CGA, Hercules kompatibel
- 256 Farben aus 262.144
- 800 x 600 Punkte, 16 Farben
- 32 Bit Video RAM
- bis zu 400% schneller als EGA
- Autosync: vollautomatische Umschaltung
- Analog- und Digital-Monitor-Anschluß
- PC/XT/AT PS/2 Mod.30 kompatibel
- 5 Jahre Garantie

VIDEO SEVEN

COMPUTER 2000 AG
Garmischer Str. 4-6
8000 München 2
Tel. 089/5 19 96-0
Zürich 042/65 11 33
Wien 02 22/85 41 02

COMPUTER 2000
Wir wissen, was läuft.

```
/*************************************************
*   test_sqrt()   Floatingpoint-Quadratwurzel!   *
 *************************************************/

test_sqrt()

{ unsigned      i;
  double        x,
                y;

  ftime( &start );

  for( i = 0; i < IMAX; i++ )
     y = sqrt( x );

  ftime( &stop );

}


/*****************************************************************
*   PRINT_RESULT   Berechnung der Differenz 'Stopzeit-Startzeit'  *
*                  und Ausgabe in µs inkl. Testtyp!              *
 *****************************************************************/

print_result( op_typ )
char    *op_typ;

{ int   millisek;

  if ( stop.time == start.time )
     millisek = stop.millitm - start.millitm;

  else
    { start.time++;
      if ( (millisek = 1000 - start.millitm + stop.millitm)  >  1000 )
         { millisek -= 1000;
           stop.time++;
         }
    }

  printf(  "\nMittlere Dauer einer %s-Operation: %u µs\n", op_typ,
           ( 1000*(stop.time-start.time)+millisek) / 50              );
}


/*************************************************
*   MAIN   Aufruf der Testroutinen, anschließende  *
*          Ausgabe der Instruktions-Laufzeit       *
 *************************************************/

main()

{ test_add();
  print_result( " FP-ADD" );

  test_mul();
  print_result( " FP-MUL" );

  test_sqrt();
  print_result( "FP-SQRT" );

}
```

```
struct timeb  {  time_t     time;
                 unsigned   short millitm;
                 short      timezone;
                 short      dstflag;          };
```

Bild 16. Der Aufbau der Zeit-Struktur „timeb"

```
>SET NO87=!80287
>FP_EXTIM

!80287

Mittlere Dauer einer  FP-ADD-Operation: 304 µs

Mittlere Dauer einer  FP-MUL-Operation: 406 µs

Mittlere Dauer einer FP-SQRT-Operation: 921 µs

>SET NO87=
>FP_EXTIM

Mittlere Dauer einer  FP-ADD-Operation: 52 µs

Mittlere Dauer einer  FP-MUL-Operation: 61 µs

Mittlere Dauer einer FP-SQRT-Operation: 118 µs
```

Bild 17. Die Laufzeit von FP_EXTIM.EXE als Funktion der Variablen NO87

CV in den Optionen entsprechend berücksichtigt, d.h. die generierte Datei name.EXE enthält alle Informationen wie Namen, Zeilennummern etc. für die anschließende symbolische „Entwanzung" (Debugging)! Eine derartige Datei ist selbstverständlich in gleicher Weise lauffähig wie ihre kompakte Form, belegt jedoch mehr Platz auf externen Speichereinheiten und ist im Code weder ablauf- noch umfangsoptimiert! Die Tabelle in *Bild 14* enthält eine Aufstellung der einzelnen Eingabe Optionen, deren Bedeutung sowie Alternativen dazu.

Das C-Programm FP_EXTIM.C aus *Bild 15* soll dazu dienen, die bisherigen Ausführungen in einer ersten Demonstration zu verdeutlichen.

Nacheinander werden hier in 'main' die drei Routinen 'test_add', 'test_mul' und 'test_sqrt' aufgerufen und danach jeweils die mittlere Zeit einer jeden Instruktion ausgegeben. Wie noch weiter unten gezeigt, dienen die Zeitwerte hier lediglich zu Vergleichszwecken und ihr absoluter Wert ist sowohl von der Taktfrequenz des verwendeten Rechners als auch der Genauigkeit der eingebauten Zeitfunktion etc. abhängig. Die Testroutinen führen jeweils 50 000mal ihre zugeordnete Instruktion sowie eine Zuweisung an eine Variable aus. Zu Beginn der „for"-Schleife und an ihrem Ende wird jeweils der aktuelle Stand der internen Uhr durch 'ftime' in den Variablen 'start' und 'stop' abgelegt. Diese Variablen sind Strukturen des Typs 'timeb' und in den „sys\"-Include-Files gemäß *Bild 16* definiert. Der dabei in der Struktur auftretende Variablentyp „time_t" ist in SYS\TYPES.H durch die Typ-Definition

typedef long time_t;

festgelegt. Die nachfolgend aufgerufene Routine 'print_time' berechnet aus den Struktur-Elementen 'time' (Anzahl Sekunden seit 1. 1. 1980) und 'millitm' (Bruchteil von 'time' in $\frac{1}{1000}$ Sekunden) die mittlere Laufzeit für eine Instruktion und gibt diesen Wert auf der Konsole aus. *Bild 17* zeigt das Protokoll einer Sitzung, während der FP_EXTIM.EXE auf einem 10 MHz-AT inklusive eingebautem NPX einmal mit und einmal ohne die NO87-Variable aufgerufen wurde. Im Durchschnitt verringerte der Numerik-Prozessor die Programm-Laufzeit auf etwa 15%. In den beiden folgenden Beispielen soll die Programmierung des NPX auf Assembler-Ebene vorgestellt werden. Ein solcher Einsatz ist dann durchaus sinnvoll, wenn es gilt, einen hohen Rechenaufwand in kürzest möglicher Zeit zu bewältigen. Sowohl Test auf Vorhandensein als auch die Software-Emu-

lationen können entfallen, dennoch sollten solche Anwendungen kurz und überschaubar gehalten sein. Ein Assembler-Modul ist immer ein wenig unübersichtlicher als ein entsprechendes in Hochsprache! Den notwendigen Praxisbezug erhalten daher die Beispiele durch die Möglichkeit der Einbindung als Funktion in die höhere Sprache C. Noch eine Bemerkung zur allgemeinen Nomenklatur: Da jeweils ein Assembler- und ein C-Programm für eine Problemstellung entwickelt werden, entstehen zwei OBJektcode-Dateien die erst der Linker zu einem EXEkutierbaren File zusammenbindet. Beide Textdateien erhalten daher einen gemeinsamen Namensteil, der je nach Typ dann unterschiedlich endet:

name_A.ASM bzw. name_C.C

Aufgrund dieser Namensgebung läßt sich der gesamte Herstellungsprozeß der jeweils gewünschten EXE-Datei durch wenige gemeinsame BATCHes, denen nur noch 'name' zu übergeben ist, einfach automatisieren! C benutzt zur Darstellung reeller Zahlen immer das doppelt genaue Format DOUBLE, d. h., ein einzelner Parameter ist 8 Bytes lang. Die Übergabe an das Unterprogramm erfolgt via Stack in Form des aktuellen Wertes, die Übernahme dagegen über einen globalen Buffer, den Floating-Point-Accu '__fac'. Dazu lädt die Subroutine diesen Accu mit dem berechneten Resultat und gibt im CPU-Register AX dessen Offset im Datensegment zurück. Das gemischte Compiler-/Assembler-Listing aus *Bild 18* zeigt beide Vorgänge anhand eines trivialen Beispiel-Programms, um die Techniken der Parameter- Übergabe und die Programmierung von Floating-Point-Instruktionen auf Assemblerebene zu erläutern. In 'main' soll einer Variablen 'inkrement' der durch die Funktion 'routine' um 1 erhöhte Wert von 3.455 zugewiesen werden. Zum Listing ein paar Erklärungen: Zu Beginn stehen die typischen Deklarationen für Code-, Daten- und Stack-Segmente eines C-Programms mit regulärem Speichermodell. Für MS-DOS ist ein ausgeführtes Programm eine reguläre Funktion, sodaß 'main' wie jedes andere Unterprogramm mit dem Bereitstellen temporären Speichers (Stackframe) beginnt: Der Basepointer BP wird auf den Stack gebracht, der Stackpointer SP nach BP kopiert und SP anschließend um die Anzahl der benötigten Bytes für lokale Variablen dekrementiert. Die 80286-Anweisung „enter NUM,LEVEL" ersetzt hier die bekannte 8086-Instruktionsfolge

```
push    bp
mov     bp,sp
sub     sp,NUM
```

```
;       Static Name Aliases
;
        TITLE   t
;       NAME    t.c

        .286c
        .287
_TEXT   SEGMENT  BYTE PUBLIC 'CODE'
_TEXT   ENDS
_DATA   SEGMENT  WORD PUBLIC 'DATA'
_DATA   ENDS
CONST   SEGMENT  WORD PUBLIC 'CONST'
CONST   ENDS
_BSS    SEGMENT  WORD PUBLIC 'BSS'
_BSS    ENDS
DGROUP  GROUP   CONST,  _BSS,   _DATA
        ASSUME  CS: _TEXT, DS: DGROUP, SS: DGROUP, ES: DGROUP
EXTRN   __fac:NEAR
EXTRN   __fltused:NEAR
_TEXT   SEGMENT
_TEXT   ENDS
CONST   SEGMENT
$T20000         DQ      0400ba3d70a3d70a4H                              ; DOUBLE-Konstante 3.455000000000000
CONST   ENDS

_TEXT   SEGMENT
;;*** #include       (math.h)
;;***
;;*** double  routine( double );
;;***
;;***
;;*** main()
;;***
;;*** {  double  inkrement = routine( 3.455 );  }                        ; inkrement := 4.455
; Line 8
        PUBLIC  _main
_main   PROC NEAR
        *** 000000      c8 08 00 00             enter   WORD PTR 8,0            ; Stackframe für 'inkrement' generieren
        *** 000004      57                      push    di
        *** 000005      56                      push    si
        inkrement = -8
        *** 000006      9b dd 06 00 00          fld     $T20000                 ; 3.455 nach ST(0) laden
        *** 00000b      83 ec 08                sub     sp,8                    ; Stack für DOUBLE-Parameter vorbereiten
        *** 00000e      8b dc                   mov     bx,sp
        *** 000010      9b dd 1f                fstp    QWORD PTR [bx]          ; ST(0) zum Stack transferieren
        *** 000013      90 9b                   fwait                           ; auf 80287 synchronisieren
        *** 000015      e8 00 00                call    _routine                ; Beispiel-Routine aufrufen
        *** 000018      83 c4 08                add     sp,8                    ; Stack vom Parameter befreien
        *** 00001b      8d 7e f8                lea     di,[bp-8]               ; Zieloffset -> 'inkrement'
        *** 00001e      8b f0                   mov     si,ax                   ; Quelloffset -> '__fac'
        *** 000020      16                      push    ss
        *** 000021      07                      pop     es                      ; Zielsegment := Stack für lokale 'inkrement'
        *** 000022      a5                      movsw
        *** 000023      a5                      movsw
        *** 000024      a5                      movsw
        *** 000025      a5                      movsw                           ; 8 Bytes von '__fac' nach 'inkrement'
        *** 000026      5e                      pop     si
        *** 000027      5f                      pop     di
        *** 000028      c9                      leave
        *** 000029      c3                      ret

_main   ENDP

;;***
;;***
;;*** double  routine( number )
;;*** double  number;
; Line 12
_TEXT   ENDS
CONST   SEGMENT
$T20001         DQ      03ff0000000000000H                              ; DOUBLE-Konstante 1.000000000000000
CONST   ENDS
_TEXT   SEGMENT
        PUBLIC  _routine
_routine        PROC NEAR
        *** 00002a      c8 00 00 00             enter   WORD PTR 0,0            ; Stackframe generieren
;;***
;;*** {  return( ++number );  }
; Line 14
;       number = 4
        *** 00002e      9b dd 06 08 00          fld     $T20001                 ; DOUBLE-Konstante 1.0 nach ST(0) laden
        *** 000033      9b dc 46 04             fadd    QWORD PTR [bp+4]        ; Parameter 'number' addieren
        *** 000037      9b dd 56 04             fstp    QWORD PTR [bp+4]        ; Resultat bei 'number'
        *** 00003b      9b dd 1e 00 00          fstp    QWORD PTR __fac         ;   und im Floating-Point-Accu ablegen
        *** 000040      b8 00 00                mov     ax,OFFSET __fac         ; Adresse des '__fac' in AX zurückgeben
        *** 000043      90 9b                   fwait                           ; auf NPX-Store synchronisieren
        *** 000045      c9                      leave
        *** 000046      c3                      ret

_routine        ENDP
_TEXT   ENDS
END
```

Bild 18. DOUBLE-Parameter-Übergabe und Übernahme

```
        mov     ax,8            ; AX := Anzahl benötigter Bytes
        call    __chkstk        ; noch genügend Platz auf dem Stack?
```

Bild 19. Automatische Stackprobe des Compilers

```
        TITLE   Mittelwert-Bildung

; ********************************************************
; *                                                      *
; *   Funktion 'mittelwert' zur Berechnung des Mittelwertes  *
; *   von 'number' Elementen eines Arrays doppelt genauer    *
; *   Real-Zahlen.                                       *
; *                                                      *
; *   ASM286 unter Verwendung des Co-Prozessors 80287 für    *
; *   den Aufruf aus C-Programmen in MS-DOS-Umgebung.    *
; *                                                      *
; *   Assemble    :  MASM MIWE_A /R;                     *
; *                                                      *
; *   Deklaration :  double mittelwert( double *, int );     *
; *                                                      *
; *   Aufruf      :  double  array[number], mw;          *
; *                                                      *
; *                  mw = mittelwert( array, number );   *
; *                                                      *
; ********************************************************

        .286c
        .287

_TEXT   SEGMENT  BYTE PUBLIC 'CODE'
_TEXT   ENDS

_DATA   SEGMENT  WORD PUBLIC 'DATA'
_DATA   ENDS

CONST   SEGMENT  WORD PUBLIC 'CONST'
CONST   ENDS

_BSS    SEGMENT  WORD PUBLIC 'BSS'
_BSS    ENDS

DGROUP  GROUP   CONST,  _BSS,   _DATA
        ASSUME  CS: _TEXT, DS: DGROUP, SS: DGROUP, ES: DGROUP

EXTRN   __fac:NEAR                      ; 8 Byte REAL-Accu für Resultat-Übergabe

_TEXT      SEGMENT

        PUBLIC  _mittelwert
_mittelwert     PROC NEAR

        enter   0,0                     ; Stackframe generieren
        push    si                      ; SI wird verwendet!

        mov     cx, WORD PTR [bp+6]     ; counter := parameter 'number'
        mov     si, WORD PTR [bp+4]     ; pointer := parameter 'array'

        fld     QWORD PTR [si]          ; ST(0)   := 1. Array-Element
        dec     cx                      ; im counter berücksichtigen

sum:    add     si,8                    ; pointer auf nächstes Array-Element
        fld     QWORD PTR [si]          ; ST(0)   := nächstes Array-Element
        fadd                            ; addieren
        loop    sum                     ; für alle Elemente wiederholen

        fidiv   WORD PTR [bp+6]         ; Summe in ST(0) / 'number'
        fstp    QWORD PTR __fac         ; Floating-Point-Accu := Mittelwert

        fwait                           ; auf NPX-Store synchronisieren
        mov     ax, OFFSET __fac        ; Adresse( FP-Accu ) zurückgeben

        pop     si
        leave                           ; Stackframe freigeben
        ret

_mittelwert     ENDP
_TEXT           ENDS

END
```

Bild 20. Die Routine Mittelwert: MIWE_A.ASM

mit NUM = 8 zur Bereitstellung eines QWORDs (Quad WORD, 8 Bytes) für 'inkrement'. Zusätzlich gibt LEVEL das lexikalische Nesting (Unterprogramm-Ebene) der Routine innerhalb des Hochsprachen-Codes und damit die Anzahl von Framepointern an, die von vorhergehenden Frames in den aktuellen zu kopieren sind. Danach kopiert der Prozessor mit Hilfe entsprechender NPX-Instruktionen die Konstante 3.455 in den Stackframe und ruft 'routine' auf. Das Unterprogramm gibt die Offsetadresse des Floating-Point-Accus zurück und der Prozessor kopiert nun dessen Inhalt in die Variable 'inkrement'. Schaut man sich den erzeugten Maschinencode einer jeden NPX-Instruktion an, so ist das durch MSC automatisch vorangestellte WAIT (09bH) auffällig. Wie weiter unten zu sehen, vollzieht der Assembler MASM die gleichen Schritte. Auf eine elegante Weise wird so in jedem Fall eine Synchronisierung der CPU mit dem NPX erreicht. Eine Ausnahme bildet die letzte Numerik-Instruktion einer Funktion. Häufig ein STORE des TOS-Registers, folgt ihr direkt die Rückkehr zur aufrufenden Routine, sodaß hier in jedem Fall ein WAIT zusätzlich einzufügen ist! Ohne diese abschließende Synchronisierung sind „zufällige" Ergebnisse keine Seltenheit; nach manchen Aufrufen hat der asynchron arbeitende NPX den Floating-Point-Accu bereits geladen, bevor sein Inhalt vom Aufrufer weiter verarbeitet wird, in manchen Fällen aber auch nicht. Der Schluß des Listings enthält die Funktion 'routine', in die in oben beschriebener Weise eingetreten wird. Auf den übergebenen Parameter 'number' kann 'routine' via BP zugreifen. C bringt alle einer Funktion übergebenen Parameter in der Reihenfolge von rechts nach links auf den Stack, so daß nach dem „enter" der am weitesten links stehende mit [bp+4] adressiert werden kann. Auf jeden weiteren, n Bytes umfassenden, Parameter kann man dann mit [bp+4+n] zugreifen. Zu beachten ist noch folgendes: Nach Assembler und Compiler-Konvention (s. [3]) ist der Name externer Subroutinen mit einem führenden Underscore-Zeichen „_" zu versehen. Für interne Zwecke reserviert der Compiler weiterhin Variablen, die mit zwei dieser Zeichen beginnen, wie z.B. „__fac". Wenn nicht genau bekannt, sollte man jedoch die Benutzung der erstgenannten Typen in C und der zuletzt genannten in Assembler vermeiden! Alle im folgenden vorgestellten Assembler-Programme sind als NEAR- Prozeduren entworfen, d. h., ihr Aufruf erfolgt aus dem gleichen CODE-Segment, in dem auch das aufrufende Programm residiert. Das eine jede Routine abschließende 'RET' berück-

sichtigt daher nur eine Offset-Adresse auf dem Stack! Bei Auslegung als FAR-Prozedur ist selbstverständlich auch ein Inter-Segment-Aufruf möglich. Dieser Fall ist dann aber auch bei der Parameter-Übergabe von Adressen zu berücksichtigen. Das hier vorausgesetzte gleiche Daten-Segment für Aufrufer und Prozedur (CPU-Register DS bleibt konstant!) ist durch die eventuell notwendige, zusätzliche Übergabe einer neuen Segment-Adresse zu modifizieren. Für die C- Ebene wird dann beispielsweise aus einem 'CHAR *' ein 'CHAR FAR *'! Weiterhin fehlt allen Prozeduren die vom Compiler zu Beginn automatisch eingefügte Stackprobe (*Bild 19*), eine Sequenz aus zwei Instruktionen, die vor Ausführung der eigentlichen Aufgaben den auf dem Stack benötigten Platz mit dem dort zur Verfügung stehenden vergleicht. Der negative Fall generiert einen kritischen MS-DOS-Fehler. Ist man sicher, daß für jeden CALL genügend Platz auf dem Stack vorhanden ist, kann man mit der Compiler-Option /Gs die Einfügung der Stackprobe unterdrükken. Der Umfang des entstehenden Codes wird verringert und seine Ausführungszeit geringfügig erhöht! *Bild 20* enthält das Listing der Funktion 'mittelwert', die 'number' Elemente des Arrays reeller Zahlen 'vektor[number]' addiert und die entstandene Summe durch 'number' dividiert. Der so berechnete Mittelwert ist selbstverständlich als reelle Zahl zurückzugeben. Der Funktion sind das Array und die Anzahl seiner Elemente zu übergeben. Arrays und Strukturen werden nicht direkt sondern vielmehr durch einen Zeiger auf deren erstes Element übergeben. Relativ dazu ist dann jedes Element adressierbar. Ein C-Programm mit einem Aufruf der Funktion ist dem Listing aus *Bild 21* zu entnehmen, zu dessen Beginn die Deklaration der externen Funktion steht. Der Aufruf erfolgt aus der 'cprintf'-Funktion für die Ausgabe auf der Konsole. Einzelheiten können den kommentierten Listings entnommen werden.

In einem weiteren Beispiel sollen alle Elemente eines Arrays reeller Zahlen mit einer Konstanten multipliziert werden. Der entsprechenden Funktion 'vecmul' aus *Bild 22* sind dazu ein Zeiger auf das Array, die Anzahl der Elemente sowie die reelle Konstante zu übergeben. 'vecmul' modifiziert das Array direkt, indem es zunächst einmal das nächste Element in den NPX lädt, dort mit der Konstanten vom Stack multipliziert und anschließend das Resultat direkt wieder in das Array schreibt. *Bild 23* zeigt ein C-Beispiel, indem alle Elemente des Arrays 'vektor[vecnum]' mit 32.768 multipliziert werden. Die „for"-Schleife gibt dann die resultierenden Werte auf der Konsole ge-

```
/*******************************************************
*   Deklaration und Aufruf 'mittelwert'                *
*                                                      *
*   Berechnet den Mittelwert der Array-Elemente von    *
*   'vektor[VECNUM]' und gibt  988.637968  aus!        *
*                                                      *
*   Compile  :  MSC MIWE_C /G2 /Zi /Od;                *
*   Link     :  LINK MIWE_C+MIWE_A /CO;                *
*   Run      :  MIWE_C                                 *
*                                                      *
 *******************************************************/

#include        <stdio.h>

#define         VECNUM  5

double  mittelwert( double *, int );            /*  Deklaration  */

double vektor[VECNUM]   =  {  2003.8635,
                                39.33333,
                               191.737,
                              2519.99901,
                               188.257      };  /*  REAL-Array   */
main()

{  cprintf( "Mittelwert = %f\r\n", mittelwert( vektor, VECNUM ) );   }
```

Bild 21. Mittelwert in C: MIWE_C.C mit Einbindung von MIWE_A.OBJ

```
        TITLE   Vektor-Multiplikation
; ************************************************************
; *                                                          *
; *   Funktion 'vecmul' zur Multiplikation von 'number'      *
; *   Elementen eines Arrays doppelt genauer Real-Zahlen     *
; *   mit einer reellen Konstanten.                          *
; *                                                          *
; *   ASM286 unter Verwendung des Co-Prozessors 80287 für    *
; *   den Aufruf aus C-Programmen in MS-DOS-Umgebung.        *
; *                                                          *
; *   Assemble    :  MASM VMUL_A /R;                         *
; *                                                          *
; *   Deklaration :  vecmul( double *, int, double );        *
; *                                                          *
; *   Aufruf      :  double  array[number], scale;           *
; *                                                          *
; *                  vecmul( array, number, scale );         *
; *                                                          *
; ************************************************************
        .286c
        .287

_TEXT   SEGMENT  BYTE PUBLIC 'CODE'
_TEXT   ENDS

_DATA   SEGMENT  WORD PUBLIC 'DATA'
_DATA   ENDS

CONST   SEGMENT  WORD PUBLIC 'CONST'
CONST   ENDS

_BSS    SEGMENT  WORD PUBLIC 'BSS'
_BSS    ENDS

DGROUP  GROUP   CONST,  _BSS,   _DATA
        ASSUME  CS: _TEXT, DS: DGROUP, SS: DGROUP, ES: DGROUP

EXTRN   __fac:NEAR                      ; 8 Byte REAL-Accu für Resultat-Übergabe
_TEXT     SEGMENT

        PUBLIC  _vecmul
_vecmul PROC NEAR

        enter   0,0                     ; Stackframe generieren
        push    si                      ; SI wird verwendet

        mov     cx, WORD PTR [bp+6]     ; counter := parameter 'number'
        mov     si, WORD PTR [bp+4]     ; pointer := parameter 'array'

mul:    fld     QWORD PTR [si]          ; ST(0)   := nächstes Array-Element
        fmul    QWORD PTR [bp+8]        ; ST(0)   := ST(0)*SCALE-Konstante
        fstp    QWORD PTR [si]          ; Array-Element ersetzen

        add     si,8                    ; pointer auf nächstes Array-Element
        loop    mul                     ; für alle Elemente wiederholen

        fwait                           ; auf NPX-Store synchronisieren

        pop     si
        leave                           ; Stackframe freigeben
        ret

_vecmul ENDP
_TEXT   ENDS

END
```

Bild 22. Skalarmultiplikation von Vektoren: VMUL_A.ASM

```
/*********************************************************
*   Deklaration und Aufruf 'vecmul'                      *
*                                                        *
*   Multipliziert alle Elemente des Arrays               *
*   'vektor[VECNUM]' mit 32.768 und gibt dann die        *
*   resultierenden Werte aus!                            *
*                                                        *
*   Compile  :  MSC VMUL_C /G2 /Zi /Od;                  *
*   Link     :  LINK VMUL_C+VMUL_A /CO;                  *
*   Run      :  VMUL_C                                   *
*                                                        *
 *********************************************************/

#include        <stdio.h>

#define         VECNUM  5

void    vecmul( double *, int, double );        /*  Deklaration  */

double vektor[VECNUM]   =   {  2003.8635,
                               39.33333,
                              191.737,
                             2519.99901,
                              188.257      };  /*  REAL-Array   */

main()

{  int  i;

   vecmul( vektor, VECNUM, 32.768 );

   for ( i = 0; i < VECNUM; i++ )
      cprintf( "vektor[%i] = %6.6f\r\n", i, vektor[i] );

}
```

Bild 23. VMUL_C.C bindet VMUL_A.OBJ ein

```
>VMUL_C
vektor[0] = 65662.599168
vektor[1] = 1288.874557
vektor[2] = 6282.838016
vektor[3] = 82575.327560
vektor[4] = 6168.805376
```

Bild 24. Das gibt VMUL_C.EXE aus

mäß *Bild 24* aus. Das letzte Beispiel zeigt anhand einer kleinen Sammlung von Routinen für komplexe Berechnungen, wie die geschickte Benutzung des NPX-Stacks sowie der Einsatz von kombinierten Instruktionen den Code-Umfang und die Ausführungszeit eines Assembler-Programms sehr gering hält. Zwischenergebnisse, die auf Hochsprache eventuell entstehen und in dafür vorgesehenen Variablen zu halten sind, können auf dieser Ebene einfach im Stack abgelegt und weiter verarbeitet werden! *Bild 25* zeigt das Listing der Datei CMPLX_A.ASM mit den vier Routinen c_add, c_sub, c_mul und c_div. Parameter-Übernahme und Übergabe erfolgt aus Zeit- und Code-Umfangsgründen in Form von Adressen, d.h, jede Routine greift in Form eines Pointers auf die Eingangs-Parameter

```
        TITLE   Komplexe Berechnungen
; ***************************************************************
; *                                                             *
; *   Die Funktionen dieses Moduls führen die vier Grundrechen- *
; *   arten in der Menge der komplexen Zahlen durch. Sie folgen *
; *   dabei den Definitionen:                                   *
; *                                                             *
; *   var1 = a + b*j,     var2 = c + d*j                        *
; *                                                             *
; *   var1 + var2  =  ( a+c )  +  ( b+d )*j                     *
; *   var1 - var2  =  ( a-c )  +  ( b-d )*j                     *
; *                                                             *
; *   var1 * var2  =  ( a*c - b*d ) + ( a*d + b*c )*j           *
; *                                                             *
; *                    a*c + b*d       b*c - a*d                *
; *   var1 / var2  =  ----------- +  ----------*j               *
; *                     c² + d²         c² + d²                 *
; *                                                             *
; *   Auf C-Ebene stellt sich eine komplexe Zahl als folgende   *
; *   Struktur dar:                                             *
; *                                                             *
; *   struct complex( double real, imag );                      *
; *                                                             *
; *   Die allgemeine Deklaration sowie der nachfolgende         *
; *   Aufruf lauten:                                            *
; *                                                             *
; *   struct complex    var1, var2, result;                     *
; *                                                             *
; *   c_XXX(struct complex*, struct complex*, struct complex* ); *
; *                                                             *
; *   c_XXX( &var1, &var2, &result );                           *
; *                                                             *
; *   Die Kommentare der Assembler-Statements benennen die      *
; *   Komponenten der beiden komplexen Eingangs-Variablen       *
; *   mit den oben beschriebenen Buchstaben a bis d!            *
; *                                                             *
; *   Mit Hilfe der beiden CPU-Register BX und SI werden die    *
; *   Adressen der beteiligten Strukturen übernommen und        *
; *   damit direkt bzw. mit einem Offset von +8 auf die         *
; *   einzelnen Realteile referiert.                            *
; *                                                             *
; *   Assemble    :  MASM CMPLX_A /R;                           *
; *                                                             *
; ***************************************************************

        .286c
        .287

_TEXT   SEGMENT  BYTE PUBLIC 'CODE'
_TEXT   ENDS

_DATA   SEGMENT  WORD PUBLIC 'DATA'
_DATA   ENDS

CONST   SEGMENT  WORD PUBLIC 'CONST'
CONST   ENDS

_BSS    SEGMENT  WORD PUBLIC 'BSS'
_BSS    ENDS
DGROUP  GROUP   CONST,  _BSS,   _DATA
        ASSUME  CS: _TEXT, DS: DGROUP, SS: DGROUP, ES: DGROUP

_TEXT   SEGMENT


        PUBLIC  _c_add
_c_add  PROC NEAR

        enter   0,0                     ; Stackframe generieren
        push    si                      ; SI wird verwendet!

        mov     bx, [bp+4]              ; pointer auf 'var1'
        mov     si, [bp+6]              ; pointer auf 'var2'

        fld     QWORD PTR [si]
        fadd    QWORD PTR [bx]          ; a + c

        fld     QWORD PTR [si+8]
        fadd    QWORD PTR [bx+8]        ; b + d

        mov     bx, [bp+8]              ; pointer auf 'result'

        fstp    QWORD PTR [bx+8]        ; 'result.imag'
        fstp    QWORD PTR [bx]          ; 'result.real'

        fwait                           ; auf NPX-Store synchronisieren

        pop     si
        leave                           ; Stackframe freigeben
        ret

_c_add  ENDP

        PUBLIC  _c_sub
_c_sub  PROC NEAR

        enter   0,0                     ; Stackframe generieren
        push    si                      ; SI wird verwendet!

        mov     bx, [bp+4]              ; pointer auf 'var1'
        mov     si, [bp+6]              ; pointer auf 'var2'
```

Bild 25. Ein Modul mit Routinen zu komplexen Funktionen

zu und legt das Resultat mit Hilfe der ebenfalls übergebenen Adresse direkt in der Ziel-Variablen ab. Für das aufrufende Programm ergibt sich hier lediglich die Notwendigkeit, drei Offset-Adressen auf den Stack zu bringen und anschließend die jeweilige Routine aufzurufen. Der Aufruf ist dadurch gleich mit der sonst notwendigen Zuweisung eines zurückgelieferten Ergebnisses verbunden; die einzige Verwaltungstätigkeit des aufrufenden Programms ist dann noch die notwendige Stack-Korrektur nach der Rückkehr aus der Routine! Den einzelnen Prozeduren liegen die im Programmkopf aufgeführten Berechnungs-Vorschriften zugrunde.

Damit bleiben c_add und c_sub sehr übersichtlich; der jeweils entstehende Realteil des Resultats verbleibt bis zur Ablage des imaginären Teils auf dem Stack und wird dann abschließend durch 'fstp QWORD PTR [bx]' bei dessen gleichzeitiger Bereinigung in der Variablen-Struktur abgelegt. Die gleichen Verhältnisse, aufgrund einer höheren Anzahl von entstehenden Zwischenergebnissen jedoch mit tieferem Stack-Level, ergeben sich bei der Multiplikation c_mul. Etwas komplizierter gestaltet sich die komplexe Division c_div. Der zunächst berechnete, Real- und Imaginär-Teil gemeinsame Nenner 'c^2+d^2' wird zweimal auf dem Stack abgelegt. Steht dann nach den folgenden Instruktionen der reelle Zähler 'a*c+b*d' als TOS zur Verfügung, führt 'fdivrp ST(2),ST' die notwendige Division in umgekehrter Richtung durch, überschreibt einen Nenner und bereinigt den Stack. Die gleichen Operation führen die nächsten Instruktionen mit dem imaginären Zähler 'b*c-a*d' durch, sodaß nach der abschließenden Instruktion 'fdivrp ST(1),ST' nur mehr die beiden Resultat-Teile auf dem Stack verbleiben! Die obligate Zuweisung an die Resultat-Variable erfolgt in der schon von den oben beschriebenen Routinen her bekannten Weise. Eine C-Anwendung für dieses Assemblerbeispiel zeigt das Listing aus *Bild 26*. Nach Definition der komplexen Variablen als Struktur zweier reeller Zahlen doppelter Genauigkeit folgt die Deklaration der externen Assembler-Routinen. Wie weiter oben beschrieben, werden ihnen die Parameter in Form von Zeigern (*) auf die jeweilige Struktur übergeben! Mit Hilfe von 'c_print' kann man auf einfache Weise eine dergestalt definierte komplexe Zahl im üblichen Format mit Real- und Imaginärteil ausgeben. Optional kann diese Ausgabe mit einem führenden Text versehen sein, auf den der Zeiger „char *mes“ weist. In 'main' schließlich werden die komplexen Variablen a und b vorbesetzt und anschließend deren Werte sowie die Resultate der vier komplexen Verknüpfungen mit ihnen auf der Konsole ausgegeben (*Bild 27*). Ausgehend von der Prämisse, einen möglichst kompakten und schnellen Code für eine numerische Anwendung mit dem NPX zu erzeugen, zeigt gerade das letzte Beispiel den Vorteil einer maschinennahen Programmierung in Assembler. Interessant ist ein Vergleich dieses Entwurfs mit dem durch MSC erzeugten für die gleiche Aufgabe. *Bild 28* zeigt in einem Teil-Mischlisting die vom Compiler bei dessen größter Optimierung erzeugten Statements für eine komplexe Division!

Wie weiter oben schon angekündigt, soll eine kurze Anwendung einer Prozessor-Steuerungs-Instruktion den Beispielteil ab-

```
        fld     QWORD PTR [si]
        fsub    QWORD PTR [bx]          ; a - c

        fld     QWORD PTR [si+8]
        fsub    QWORD PTR [bx+8]        ; b - d

        mov     bx, [bp+8]              ; pointer auf 'result'

        fstp    QWORD PTR [bx+8]        ; 'result.imag'
        fstp    QWORD PTR [bx]          ; 'result.real'

        fwait                           ; auf NPX-Store synchronisieren

        pop     si
        leave                           ; Stackframe freigeben
        ret

_c_sub  ENDP
        PUBLIC  _c_mul
_c_mul  PROC NEAR

        enter   0,0                     ; Stackframe generieren
        push    si                      ; SI wird verwendet!

        mov     bx, [bp+4]              ; pointer auf 'var1'
        mov     si, [bp+6]              ; pointer auf 'var2'

        fld     QWORD PTR [bx]
        fmul    QWORD PTR [si]          ; a*c

        fld     QWORD PTR [bx+8]
        fmul    QWORD PTR [si+8]        ; b*d

        fsub                            ; a*c - b*d

        fld     QWORD PTR [bx]
        fmul    QWORD PTR [si+8]        ; a*d

        fld     QWORD PTR [bx+8]
        fmul    QWORD PTR [si]          ; b*c

        fadd                            ; a*d + b*c

        mov     bx, [bp+8]              ; pointer auf 'result'

        fstp    QWORD PTR [bx+8]        ; 'result.imag'
        fstp    QWORD PTR [bx]          ; 'result.real'

        fwait                           ; auf NPX-Store synchronisieren

        pop     si
        leave                           ; Stackframe freigeben
        ret

_c_mul  ENDP

        PUBLIC  _c_div
_c_div  PROC NEAR

        enter   0,0                     ; Stackframe generieren
        push    si                      ; SI wird verwendet!

        mov     bx, [bp+4]              ; pointer auf 'var1'
        mov     si, [bp+6]              ; pointer auf 'var2'

        fld     QWORD PTR [si]
        fmul    ST,ST                   ; c²

        fld     QWORD PTR [si+8]
        fmul    ST,ST                   ; d²

        fadd                            ; c² + d²
        fld     ST                      ; ( c² + d² )  duplizieren

        fld     QWORD PTR [bx]
        fmul    QWORD PTR [si]          ; a*c

        fld     QWORD PTR [bx+8]
        fmul    QWORD PTR [si+8]        ; b*d

        fadd                            ; a*c + b*d
        fdivrp  ST(2),ST                ; (a*c+b*d)/(c²+d²) & POP!

        fld     QWORD PTR [bx+8]
        fmul    QWORD PTR [si]          ; b*c

        fld     QWORD PTR [bx]
        fmul    QWORD PTR [si+8]        ; a*d

        fsub                            ; b*c - a*d
        fdivrp  ST(1),ST                ; (b*c-a*d)/(c²+d²) & POP!

        mov     bx, [bp+8]              ; pointer auf 'result'

        fstp    QWORD PTR [bx+8]        ; 'result.imag'
        fstp    QWORD PTR [bx]          ; 'result.real'

        fwait                           ; auf NPX-Store synchronisieren

        pop     si
        leave                           ; Stackframe freigeben
        ret

_c_div  ENDP

_TEXT   ENDS

END
```

```
/*******************************************************
*   Deklaration und Aufruf 'cmplx_a.obj'               *
*                                                      *
*   Initialisiert zwei komplexe Variablen und gibt     *
*   deren Werte sowie Verknüpfungs-Resultate aus!      *
*                                                      *
*   Compile  :  MSC CMPLX_C /G2 /Zi /Od;               *
*   Link     :  LINK CMPLX_C+CMPLX_A /CO;              *
*   Run      :  CMPLX_C                                *
*                                                      *
 *******************************************************/

#include        <stdio.h>

struct    complex     {  double    real, imag;    };

void      c_add( struct complex*, struct complex*, struct complex* );
void      c_sub( struct complex*, struct complex*, struct complex* );
void      c_mul( struct complex*, struct complex*, struct complex* );
void      c_div( struct complex*, struct complex*, struct complex* );


c_print( mes, c_num )       /*  Gibt komplexe Zahl aus   */
char    *mes;
struct    complex   c_num;

{  cprintf( "%s(%.6f) + (%.6f)j\r\n",
            mes, c_num.real, c_num.imag    );
}
```

```
main()

{ struct  complex   a, b, c;

  a.real = 100.5;
  a.imag = 10.75;

  b.real = 99.5;
  b.imag = 9.25;

  c_print( "var1         = ", a );
  c_print( "var2         = ", b );
  cprintf( "\r\n" );

  c_add( &a, &b, &c );
  c_print( "var1 + var2 = ", c );

  c_sub( &a, &b, &c );
  c_print( "var1 - var2 = ", c );

  c_mul( &a, &b, &c );
  c_print( "var1 * var2 = ", c );

  c_div( &a, &b, &c );
  c_print( "var1 / var2 = ", c );
}
```

Bild 26. CMPLX_A.OBJ ist in CMPLX_C.C eingebunden

```
>CMPLX_C
var1         = (100.500000) + (10.750000)j
var2         = (99.500000) + (9.250000)j

var1 + var2 = (200.000000) + (20.000000)j
var1 - var2 = (-1.000000) + (-1.500000)j
var1 * var2 = (9900.312500) + (1999.250000)j
var1 / var2 = (1.011354) + (0.014020)j
```

Bild 27. Das gibt CMPLX_C.EXE aus

```
;       Static Name Aliases
;
        TITLE   t
        NAME    t.C      Compile:  MSC T /G2 /Ox /Fc;
;
;*** struct  complex {  double   real, imag;  };
;***
;***
;***
;***
;*** c_div( var1, var2, result )
;*** struct  complex *var1, *var2, *result;
; Line 7
        PUBLIC  _c_div
_c_div  PROC NEAR
        *** 000000      c8 10 00 00     enter   WORD PTR 16,0
        *** 000004      57              push    di
        *** 000005      56              push    si
;***
;*** { double temp;
; Line 9
;       var1 = 4
;       var2 = 6
;       result = 8
;       temp = -8
;***
```

```
;***   (*result).real = ( (*var1).real*(*var2).real + (*var1).imag*(*var2).imag ) /
;***                    ( temp = (*var1).real*(*var1).real + (*var1).imag*(*var1).imag ) ;
; Line 12
        *** 000006      8b 46 04                mov     ax,[bp+4]               ; var1
        *** 000009      8d 7e f0                lea     di,[bp-16]
        *** 00000c      8b f0                   mov     si,ax
        *** 00000e      16                      push    ss
        *** 00000f      07                      pop     es
        *** 000010      a5                      movsw
        *** 000011      a5                      movsw
        *** 000012      a5                      movsw
        *** 000013      a5                      movsw
        *** 000014      8b 5e 08                mov     bx,[bp+8]               ; result
        *** 000017      8b f0                   mov     si,ax
        *** 000019      9b dd 44 08             fld     QWORD PTR [si+8]
        *** 00001d      8b 76 06                mov     si,[bp+6]               ; var2
        *** 000020      9b dc 4c 08             fmul    QWORD PTR [si+8]
        *** 000024      9b dd 04                fld     QWORD PTR [si]
        *** 000027      9b dc 4e f0             fmul    QWORD PTR [bp-16]
        *** 00002b      9b de c1                fadd
        *** 00002e      8b f0                   mov     si,ax
        *** 000030      9b dd 44 08             fld     QWORD PTR [si+8]
        *** 000034      9b dc 4c 08             fmul    QWORD PTR [si+8]
        *** 000038      9b dd 46 f0             fld     QWORD PTR [bp-16]
        *** 00003c      9b dc 4e f0             fmul    QWORD PTR [bp-16]
        *** 000040      9b de c1                fadd
        *** 000043      9b dd 56 f8             fst     QWORD PTR [bp-8]        ; temp
        *** 000047      9b de f9                fdiv
        *** 00004a      9b dd 1f                fstp    QWORD PTR [bx]
;***
;***   (*result).imag = ( (*var1).imag*(*var1).real - (*var1).real*(*var2).imag ) /
;***                    temp   ;
; Line 15
        *** 00004d      9b dd 44 08             fld     QWORD PTR [si+8]
        *** 000051      9b dc 4e f0             fmul    QWORD PTR [bp-16]
        *** 000055      8b 76 06                mov     si,[bp+6]               ; var2
        *** 000058      9b dd 44 08             fld     QWORD PTR [si+8]
        *** 00005c      9b dc 4e f0             fmul    QWORD PTR [bp-16]
        *** 000060      9b de e9                fsub
        *** 000063      9b dc 76 f8             fdiv    QWORD PTR [bp-8]        ; temp
        *** 000067      9b dd 5f 08             fstp    QWORD PTR [bx+8]
        *** 00006b      90 9b                   fwait
;***
;***
;*** }
; Line 18
        *** 00006d      5e                      pop     si
        *** 00006e      5f                      pop     di
        *** 00006f      c9                      leave
        *** 000070      c3                      ret

_c_div  ENDP
;***
;***
;***
;***
```

Bild 28. Komplexe Division, wie sie der Compiler herstellt

```
; ********************************************************
; *   Assemble    :  MASM ENV87_A /R;                    *
; *                                                      *
; *   Deklaration :  void  get_env_87( struct env87* );  *
; *                                                      *
; *   Aufruf      :  get_env_87( &e );                   *
; *                                                      *
; ********************************************************

        .286c
        .287

_TEXT   SEGMENT  BYTE PUBLIC 'CODE'
_TEXT   ENDS

_DATA   SEGMENT  WORD PUBLIC 'DATA'
_DATA   ENDS

CONST   SEGMENT  WORD PUBLIC 'CONST'
CONST   ENDS

_BSS    SEGMENT  WORD PUBLIC 'BSS'
_BSS    ENDS

DGROUP  GROUP   CONST,  _BSS,   _DATA
        ASSUME  CS: _TEXT, DS: DGROUP, SS: DGROUP, ES: DGROUP


_TEXT   SEGMENT

        PUBLIC  _get_env_87
_get_env_87  PROC NEAR

        enter   0,0                 ; Stackframe generieren

        mov     bx, [bp+4]
        fstenv  [bx]                ; 80287-Registerinhalt holen
        fwait                       ; auf NPX-Store synchronisieren

        leave                       ; Stackframe freigeben
        ret

_get_env_87  ENDP

_TEXT   ENDS
END
```

Bild 29. ENV87_A.ASM: NPX-Flags und Pointer holen

```
/*********************************************************
*   Compile   :  MSC ENV87_C /G2 /Zi /Od;                *
*   Link      :  LINK ENV87_C+ENV87_A /CO;               *
*   Run       :  ENV87_C                                 *
*                                                        *
 ********************************************************/
struct  s1      { unsigned      im    : 1;
                  unsigned      dm    : 1;
                  unsigned      zm    : 1;
                  unsigned      om    : 1;
                  unsigned      um    : 1;
                  unsigned      pm    : 1;
                  unsigned      res1  : 2;
                  unsigned      pc    : 2;
                  unsigned      rc    : 2;
                  unsigned      ic    : 1;
                  unsigned      res2  : 3;      };

struct  s2      { unsigned      ie    : 1;
                  unsigned      de    : 1;
                  unsigned      ze    : 1;
                  unsigned      oe    : 1;
                  unsigned      ue    : 1;
                  unsigned      pe    : 1;
                  unsigned      res1  : 1;
                  unsigned      es    : 1;
                  unsigned      c0    : 1;
                  unsigned      c1    : 1;
                  unsigned      c2    : 1;
                  unsigned      sp    : 3;
                  unsigned      c3    : 1;
                  unsigned      b     : 1;      };

struct  env87   { struct s1     control;
                  struct s2     status;
                  unsigned      tag;
                  unsigned      ip_l;
                  unsigned      i_op  : 11;
                  unsigned      res1  :  1;
                  unsigned      ip_h  :  4;
                  unsigned      dp_l;
                  unsigned      res2  : 12;
                  unsigned      dp_h  :  4;     };

void    get_env_87( struct env87* );
main()

{ struct   env87   e;

  get_env_87( &e );

  cprintf( "Condition-Code     ( C3 C2 C1 C0 ) : %u %u %u %u \r\n",
           e.status.c3, e.status.c2, e.status.c1, e.status.c0    );

  cprintf( "Stackpointer       ( SP          ) : %u \r\n",
           e.status.sp                                          );

  cprintf( "Opcode                             : %X \r\n",
           e.i_op                                               );

  cprintf( "Instruction-Pointer                : %X%X \r\n",
           e.ip_h, e.ip_l                                       );
}
```

Bild 30. ENV87_C.C benutzt ENV87_A.OBJ

schließen. In einem C-Programm ist der aktuelle Inhalt der Flag- und Pointer- Register des NPX in Form einer entsprechenden Variablen-Struktur für eine weitere Verarbeitung bereitzustellen. *Bild 29* enthält im Listing dazu die Assembler-Routine '_get_env_87', die in bekannter Weise eine Adresse vom Aufrufer entgegennimmt um danach mit 'fstenv [bx]' die 14 Bytes aus dem „Innern" des NPX dorthin zu transferieren. Wie *Bild 30* in einem C-Beispiel zeigt, werden diese 14 Bytes durch die Struktur 'env87' beschrieben und mit der Variablen 'e' dieses Struktur-Typs entsprechender Speicherplatz reserviert. 'env87' enthält für das CONTROL- und das STATUS-Wort jeweils eine eigene Struktur 'sX', die dem Layout dieser Worte (s. Bilder 11 und 9 des ersten Teils der Serie) entspricht. Der hier verwendete Compiler MSC beginnt beim Ablegen von Bit-Feldern mit dem niederwertigsten Bit 0 in Richtung Bit 7 und steigenden Byte-Adressen! Nach dem Aufruf von 'get_env_87' stehen die aktuellen Werte der internen NPX-Register durch einfache Nennung der Struktur-Komponenten zur Verfügung. *Bild 31* zeigt die durch dieses Beispiel erzeugte Ausgabe nach vorherigem Aufruf der weiter oben beschriebenen Mittelwert-Routine. Der Condition-Code „0 0 0 0" weist das Ergebnis der letzten Berechnung als gültig aus (s. auch Bild 10), 'mittelwert' selbst verließ den NPX mit dem Stackpointer-Wert ST=0! Der letzte abgearbeitete Opcode war 033CH oder, um die konstanten Führungsbit „11011" zur Kennzeichnung von NPX-Instruktionen erweitert (s. auch Bild 6 des ersten Teils der Serie), 0DB3CH. Ein Blick in die Dekodierungs-Tabelle [2] bestätigt, daß es sich dabei um ein 'fstp', der letzten numerischen NPX- Instruktion von 'mittelwert' handelt; das danach noch folgende 'fwait' dient ja lediglich der Synchronisierung und beeinflußt die Prozessor-Umgebung nicht mehr! Abschließend läßt sich festhalten, daß der Einsatz des Numerik- Prozessors 80287 dem AT grundsätzlich eine höhere Rechengeschwindigkeit verleiht. Vor allem in zahlenintensiven Anwendungen, wie grafischen Berechnungen in CAD-Programmen, Meßwert-Analysen in Realzeit etc. ist die Benutzung dieses heute noch relativ kostspieligen Chips fast schon ein „Muß"! Bei dem allgemein in der Elektronik-Branche zu beobachtenden Preisverfall ist es jedoch wohl nur noch eine Frage recht kurzer Zeit, wann jeder heutige AT-Besitzer seinen Rechen-Applikationen (40!) „Beine macht". Denn sicherlich macht es sich auch sehr gut, wenn Tabellenkalkulationen, Berechnungen von Sternenbahnen und manch' anderes „Rechen-Spiel" einmal in Computer-typischer Manier erledigt werden: Nämlich während eines „Fingerschnipps"!

```
>MIWE_C
Mittelwert = 988.637968

>ENV87_C
Condition-Code  ( C3 C2 C1 C0 ) : 0 0 0 0
Stackpointer    ( SP          ) : 0
Opcode                          : 33C
Instruction-Pointer             : 0FCEE
```

Bild 31. Das liefert ENV87_C.EXE

Literatur

[1] COMPUTER, March 1981, IEEE.
[2] SAB 80286 Programmer's Reference Manual.
[3] Microsoft C Compiler, User's Guide.
[4] Microsoft Macro Assembler, User's Guide.

Werner Hilf

Der MC68030

Teil 2: Software und Cache

Im ersten Teil dieser Serie wurden die Unterschiede in der Hardware des 68020 und 68030 dargestellt. Wie sieht es nun mit der Software aus? Wie weit geht die Kompatibilität und was bringen die beiden Cache-Speicher? Fragen, die im folgenden Teil des Beitrags beantwortet werden.

Das wichtigste: gravierende Software-Unterschiede zum 68020 gibt es nicht. Die Änderung bzw. Ergänzung wurde durch die zusätzliche MMU hervorgerufen. Zwei Befehle existieren beim 68030 nicht mehr: CALLM und RTM. Diese Befehle beziehen sich beim 68020 auf den Zugriffsschutz-Mechanismus.

Der 68000 erlaubt zwei Privileg-Ebenen (User und Supervisor). Im Zusammenspiel mit dem CALLM- und RTM-Befehl und der externen PMMU MC68851 lassen sich diese Ebenen erweitern – ein Prinzip aus der Großrechnertechnik. Da dieses Konzept nur von wenigen 68020-Kunden ausgenützt wurde, entschloß man sich bei Motorola, im Zuge der Transistoreinsparung diesen Zugriffsschutz beim 68030 nicht mehr zu implementieren. Daher sind diese Befehle der Rationalisierung zum Opfer gefallen. Die Kompatibilität ist insofern gewahrt, daß diese Befehle beim 68030 auf die Exception „nicht implementierter Befehl" [1] laufen; durch Erweiterung des Exception-Programmes und entsprechende Hardware können die Befehle bei Bedarf emuliert werden.

Vier neue Befehle, die der Programmierung bzw. Steuerung dienen, ergänzen den Befehlssatz des 68030. Da die PMMU-Register im Supervisor-Bereich liegen, sind diese Befehle privilegiert, sie werden im Abschnitt „Speicherverwaltungseinheit" beschrieben.

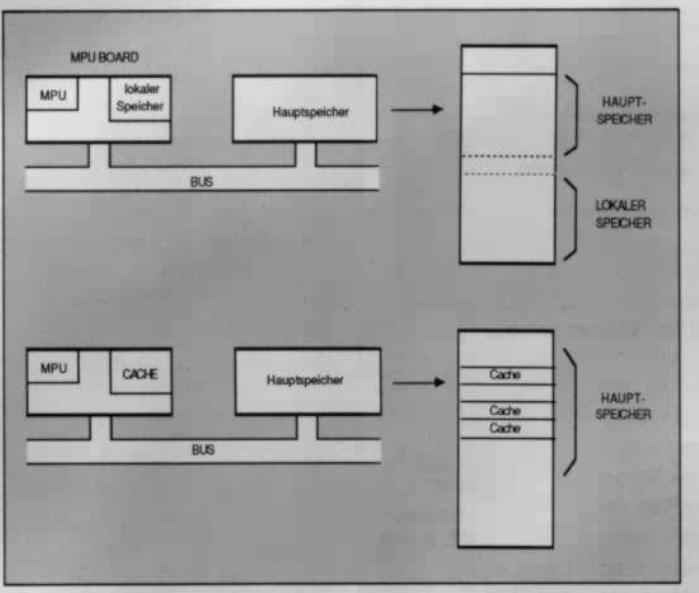


Bild 1. Lokaler Speicher und Cache: Während der erste über einen eigenen Adreßbereich angesprochen wird, liegt der Cache parallel zum Hauptspeicher und ist damit „unsichtbar"

Cache-Speicher

Eine direkte technische Übersetzung des Begriffes „Cache" gibt es nicht, man muß ihn umschreiben. Verwechslungen mit dem Ausdruck „Cash", der genauso ausgesprochen wird und im Finanzbereich angesiedelt ist, sollten vermieden werden, denn sie haben wenig gemeinsam. Cache bedeutet „versteckt aufbewahren". Was wird versteckt und warum? Es können sowohl Befehle als auch Daten sein. Die Cache-Technologie ist bereits seit Ende der 60er Jahre bekannt und im Einsatz. Auch sie verdanken wir der Computertechnik. Die Neuerung beim 68020 war der sogenannte „On-Chip-Cache". Frühere Caches waren grundsätzlich extern aufgebaut. Caches sind ein Kompromiß zwischen einem schnellen, damit teuren und langsamen, d. h. preisgünstigen externen Speicher und tragen zur Verbesserung des Preis/Leistungs-Verhältnisses eines Computersystems bei – so wurden beispielsweise durch die Einführung eines verhältnismäßig kleinen Caches (256 Byte) beim 68020 Leistungssteigerungen von 10...20% erreicht.

Ein Cache ist ein sehr schnelles RAM mit zwei charakteristischen Eigenschaften:

1. er liegt parallel zum Programm- bzw. Datenspeicher;
2. er speichert nicht nur die Information, sondern auch die Adresse, unter der diese Information abgelegt ist.

Bild 1 zeigt eine Gegenüberstellung von lokalem Speicher und Cache. Während die Adressen des lokalen Speichers und des Hauptspeichers jeweils nur einmal im System auftreten, liegt der Cache immer parallel zum Hauptspeicher und/oder zum eventuell vorhandenen lokalen Speicher. Der lokale Speicher kann nur explizit vom Anwender adressiert werden. Der Cache wird dagegen – dem Programmablauf folgend – nur von der CPU beschrieben.

Der Ablauf

Ein Programmablauf mit Befehls-Cache sieht daher folgendermaßen aus: Holt der Prozessor einen Befehl, so wird verglichen, ob der Inhalt der betreffenden Adresse – also der gesuchte Befehl – bereits im Cache steht. Caches benötigen daher einen oder mehrere Komperatoren um herauszufinden, ob der Befehl unter der Adresse im Cache vorliegt. Solche Speicher werden Assoziativspeicher genannt. Wird die gesamte Adresse verglichen, spricht man von „Voll-assoziativ-Speichern". Beim Daten- und Befehlscache des 68030 wird jedoch nur ein Teil der Adresse verglichen (*Bild 2*), daher wird dieses Cache auch „Teil-assoziativ-Speicher" genannt. Ist der Befehl im Cache, dann haben wir einen sogenannten „Hit" (Treffer): Der Prozessor muß nicht auf den „langsamen" Programmspeicher (Hauptspeicher) zugreifen, sondern kann den Befehl aus dem schnellen Cache ausführen. Steht der Befehl noch nicht im Cache, dann haben wir einen sogenannten „Miss" (Fehlgriff) und die CPU muß den Befehl aus dem Hauptspeicher holen. Während der Befehl geholt wird, schreibt ihn der Prozessor parallel dazu in den Cache. Trifft die CPU nun noch einmal auf den Befehl unter jener Adresse, so ergibt das

NETZWERK-Kommunikation

Acer LAN 30 von Multitech

ARCNet-/EtherNet-Standard
mit Novell Software 86 und 286

••• unsere PC s als Workstations und Filserver für diese Netzwerke!

Der **Acer 500** („VOLKS-PC") hat sich vieltausendfach bewährt in den unterschiedlichsten Einsatzmöglichkeiten. Unser schnellster PC auf der Basis des 8088 ist der **Acer 710** mit einer Taktgeschwindigkeit von **10** MHz. Für preisbewußte Anwender der 80286-Technik bietet sich der **Acer 910** an. Weit in die Zukunft reichen die Möglichkeiten des **Acer 900** mit seiner großen Ausbaufähigkeit im AT-Bereich. Den vorläufigen Höhepunkt der PC Erfolgsmodelle stellt der 32-bit **Acer 1100** mit dem Hochleistungs-Mikroprozessor Intel **80386** und einer Taktgeschwindigkeit von 16 bis 4,77 MHz dar.
Neben diesen gewinnbringenden Computern vertreibt **CE-TEC** das gesamte Spektrum an Zubehör und möglichen Erweiterungen von verschiedenen Herstellern, um optimale Nutzung zu gewährleisten. Grafik-, Funktions- und Spezialkarten, Tastaturen und Massenspeicher gehören zum Lieferprogramm ebenso wie **Monitore,** Netzwerke **(LAN)** und der **Laser-Drucker Acer LP-75** Das komplette Programm aus einer Hand! Etwa 300 CE-TEC-Fachhändler in der Bundesrepublik und West-Berlin geben gern weitere Auskünfte.

Acer LAN 30

DM 9.995,– (unverbindliche Preisempfehlung)

Für EtherNet Starterkit
komplett mit Software
für 3 Stationen.
(CSMA/CD - IEE 802.3 –)
10 MBit pro Sekunde,
30 Teilnehmer pro Segment

CE-TEC International

Generalimporteur:
CE-TEC Trading GmbH · Kornkamp 4 · D-2070 Ahrensburg
Telefon: 04102/49 01-0 · Telex: 2 189 875 · Telefax: 04102/49 01 38

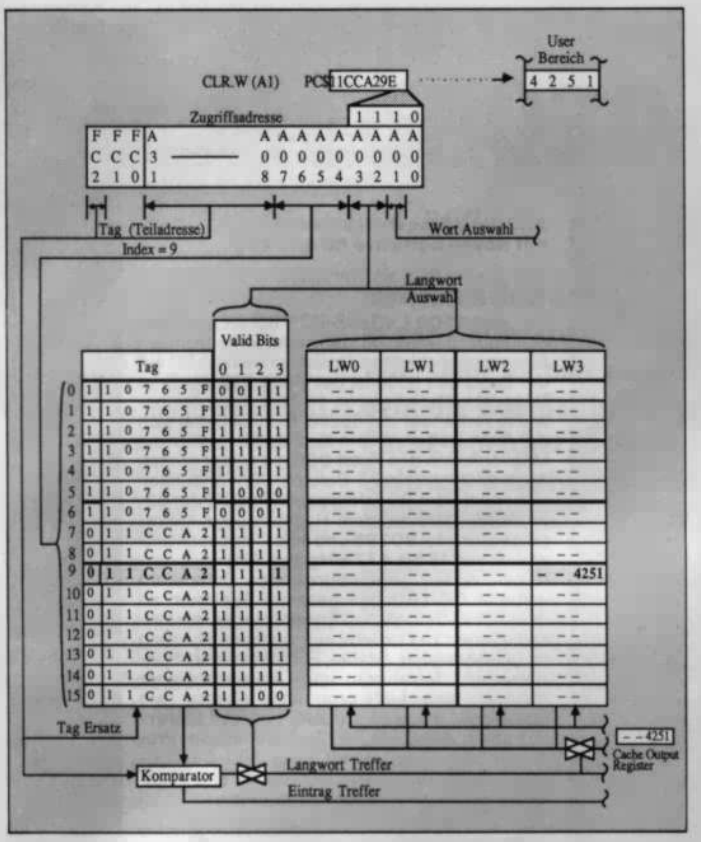


Bild 2. Im Befehlscache des 68030 wird auch die Funktionscode-Leitung FC2 ausgewertet (für Treffer entscheidenden Felder unterlegt)

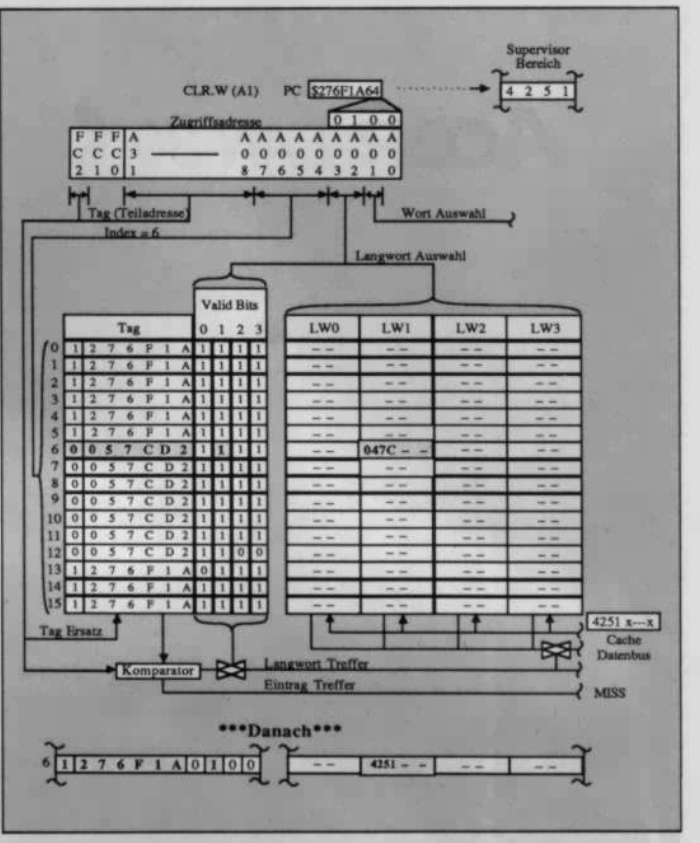


Bild 3. Hier sind die für einen Fehlgriff im Befehscache entscheidenden Felder unterlegt

einen Treffer, der Befehl kann mit einer größeren Geschwindigkeit als vorher abgearbeitet werden. Ein Cache ist demnach nur bei wiederholbaren Sequenzen sinnvoll, die höchstens so groß sind wie der Cache selbst, sonst werden die Werte jedesmal überschrieben. Sprünge außerhalb der Adreßdistanz des Caches führen ebenfalls zu einem Überschreiben des Caches. Eine geschickte Manipulation des Caches, die nur über das Cache Control Register (CACR) und das Cache Adress Register (CAAR) möglich ist, kann jedoch auch diesen Nachteil in großem Maße wettmachen. Das CACR erlaubt das Sperren (E), Einfrieren (F) und Löschen des gesamten Caches (C) oder eines Eintrages (CE). Letzteres funktioniert im Zusammenspiel mit dem CAAR. Einfrieren bedeutet hier, daß sich der Cache wie ein ROM verhält: Es werden zwar Informationen verglichen und bei Treffern gelesen, aber nicht überschrieben. Wir unterscheiden folgende Cache-Typen:

1. externer Cache
2. interner Cache (On-Chip-Cache)
3. Befehlscache
4. Datencache
5. logischer Cache
6. physikalischer Cache

Ein externer Cache bringt Vorteile, wenn beim Zugriff auf den Hauptspeicher Wartezyklen eingefügt werden müssen. Er kann jedoch nie schneller sein als ein externer Speicherzugriff. Kann der gesamte Hauptspeicher ohne Wartezyklen angesprochen werden, ist demnach kein externer Cache nötig.

Hier bringt ein interner Cache Vorteile: Zugriffe hierauf sind üblicherweise schneller als auf externe Speicher. Beim 68020 dauert zum Beispiel ein minimaler Speicherzugriff drei, ein Cache-Zugriff dagegen nur zwei Taktzyklen.

Der Unterschied zwischen logischen und physikalischen Caches wurde im ersten Teil dieser Serie dargestellt.

Der Befehls-Cache

Ein Befehlscache unterscheidet sich vom Datencache dadurch, daß lediglich Befehle in den Cache übernommen werden und nicht die daraus resultierenden Daten. Dazu ein Beispiel:

```
CLR.W (A1)
```

In den Befehlscache wird der Opcode und die möglichen (bei diesem Befehl nicht vorhandenen) Erweiterungsworte übernommen, nicht aber die mit A1 adressierte Speicherstelle samt Inhalt.

Bild 2 zeigt den Aufbau des Befehlscache im 68030 und gleichzeitig das genannte Beispiel. Wie beim 68020 haben wir einen 256 Byte großen Cache. Er wurde jedoch beim 68030 etwas umorganisiert: Während beim 68020 die Speicherung von 64 Einträgen mit 32-Bit-Werten (64 x 1 Langwort) möglich ist, erlaubt der 68030 die Ablage von 16 Einträgen mit je 128 Bit (16 x 4 Langworte). Für jedes Langwort muß ein Valid-Bit (V) verfügbar sein. Für den Anwender hat diese Umorganisation den Vorteil, daß dadurch der im Abschnitt „Hardware" beschriebene Burst-Fill-Modus angewandt und Befehle bzw. Daten noch schneller übertragen werden können. Der Nachteil: Bei jedem neuen Eintrag müssen vier Langworte ersetzt werden, denn die übrigen Valid-Bits werden gelöscht (*Bild 5*).

In Bild 2 wird ein Treffer dargestellt. Die Funktionscode-Leitung FC2 (hier: User-Modus) und die Teiladresse (englisch Tag = Kennzeichen) stimmen mit der Zugriffsadresse überein. Des weiteren ist das Valid-Bit des entsprechenden Langwortes (Bit 3) gesetzt. Der Inhalt des Langwortes 3 ist demnach identisch mit dem Inhalt (in diesem Fall dem Opcode $4251) der Zugriffsadresse $11CCA29E. Voraussetzung für einen Treffer ist selbstverständlich die Freigabe des Caches.

Bild 3 zeigt einen Cache-Fehlgriff. Weder die Teiladresse (A8...A31) noch das FC2-Bit stimmen mit dem Eintrag 6 überein, d. h. dieser Befehl steht auf einer anderen Adres-

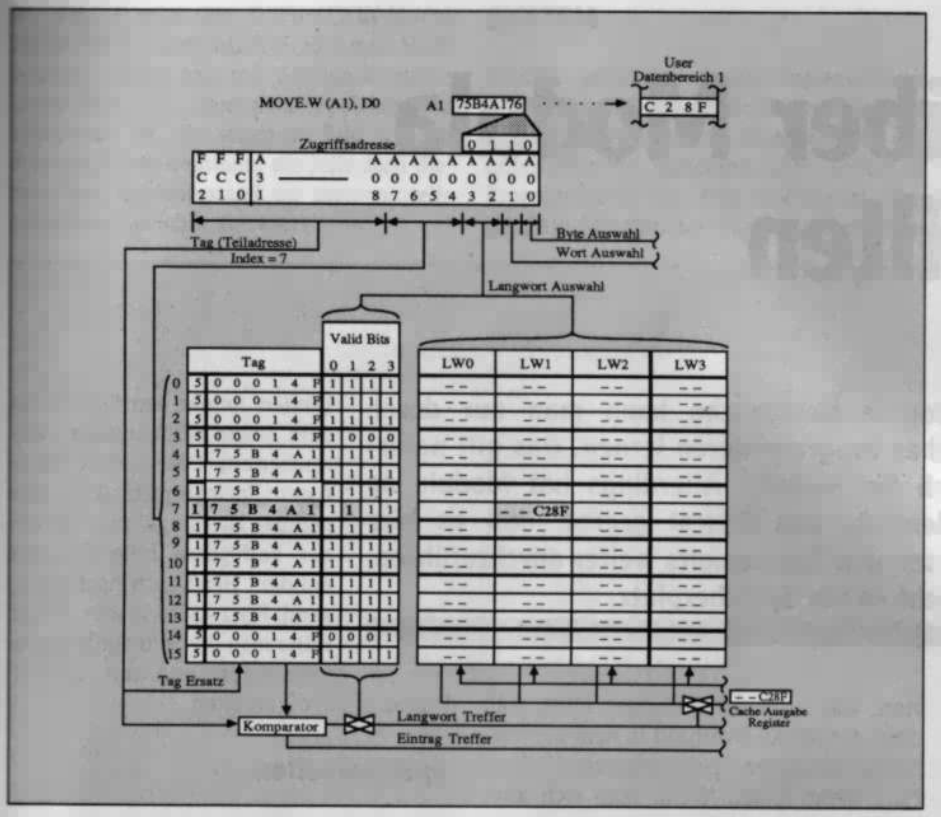


Bild 4. Im Datencache – hier bei einem Treffer – werden alle Funktionscodes (FC0...FC2) ausgewertet

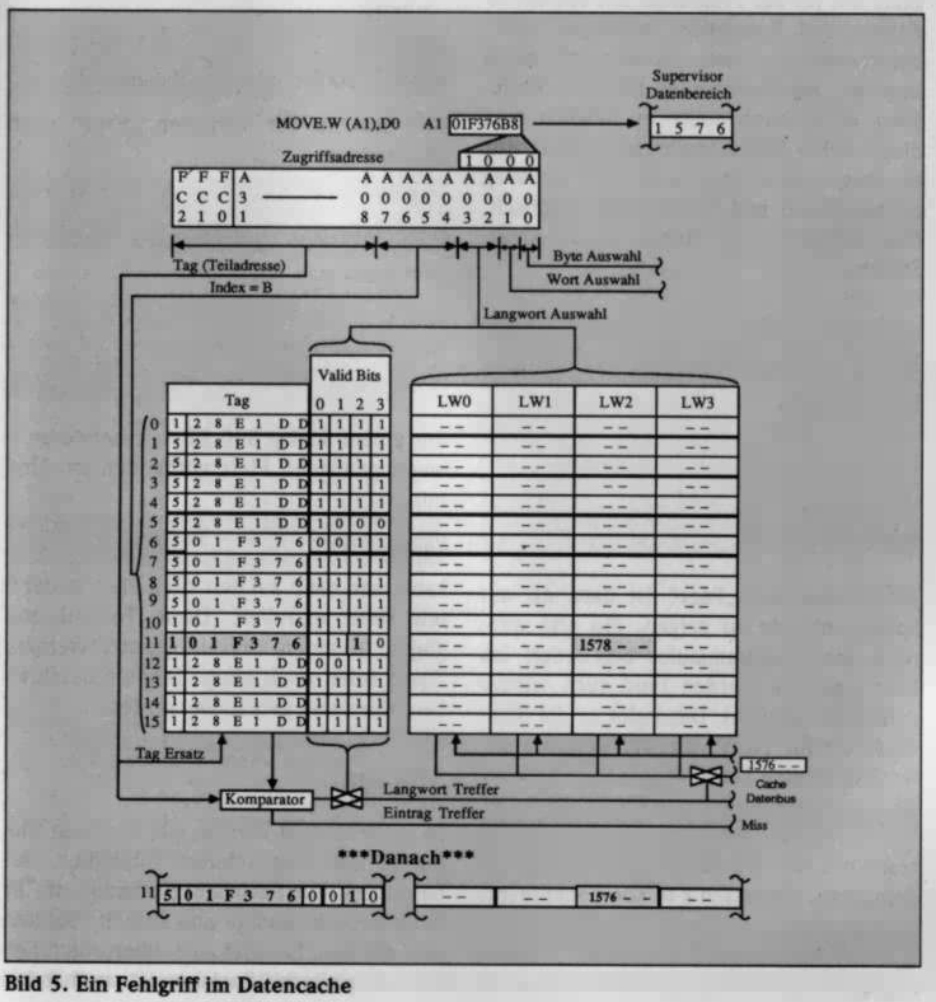


Bild 5. Ein Fehlgriff im Datencache

se im Supervisor-Bereich. Ist der Cache freigegeben und nicht eingefroren, so werden der neue Wert (Opcode $4251), die Teiladresse und das FC2-Bit aktualisiert.

Der Datencache

In *Bild 4* sehen wir den Aufbau des Datencaches im 68030. Er hat dieselbe Organisation wie der Befehlscache. Im Gegensatz zum Befehlscache sind alle Funktionscode-Bits gespeichert, um die Trennung zwischen Anwender- und Supervisor-Bereichen ausnützen zu können. Das Beispiel in Bild 4 zeigt einen Treffer, da Funktionscode und Teiladresse mit den Werten im Cache übereinstimmen. Der Datencache muß außerdem freigegeben und das Valid-Bit für das entsprechende Langwort gültig (gesetzt) sein.

Bild 5 zeigt einen Datencache-Fehlgriff für einen Lesezyklus. Die Teiladresse ist zwar identisch, aber es wird aus einem Datenbereich (FC0...FC2 = 101) zugegriffen. Ist der Datencache freigegeben und nicht eingefroren, so wird das entsprechende Datum im Cache mit dem aus dem Speicher gelesenen Wert überschrieben. Die restlichen Langworte des Eintrages werden ungültig gemacht (gelöscht). Ist der Burst-Fill-Modus freigegeben, so versucht der 68030, die ungültigen Einträge zu füllen.

Bei Schreibzyklen gelten für den Datencache besondere Regeln: Solange die Daten nur von einem Prozessor verändert werden, gibt es keine Probleme. Werden die Daten aber durch einen anderen Baustein (z. B. eine weitere CPU oder einen DMA-Controller) verändert, dann ist der Datencache nicht immer auf dem neuesten Stand. Man nennt dies „stale data (stabile Daten) problem": Im Falle eines Treffers würde der Prozessor dann alte Daten verwenden. Dieses Problem taucht üblicherweise beim Befehlscache nicht auf, denn der Programmcode steht fest und sollte im Normalfall nicht verändert werden – es sei denn, man läßt selbstmodifizierenden Code zu (dessen aktuelle Version dann im Datencache steht). In der Testphase (z. B. beim Setzen von Software-Breakpoints) ist das zum Teil unumgänglich, in der Endphase muß es verboten werden. Derartige Probleme verhindert die sogenannte „No-Write-Allocate Policy". Das bedeutet, daß bei einem Fehlgriff der Datencache nicht auf den neuesten Stand gebracht wird. Damit werden immer die aktuellen Daten geschrieben. Dieser Modus wird im CACR durch das WA-Bit programmiert und kann dynamisch von Befehls- zu Befehlszyklus geändert werden.

In der nächsten mc geht es um die Speicherverwaltungseinheit.

Ulrich Rohde

Was Sie über Modula wissen sollten

Teil 4

Eine Programmiersprache kann man nur durch praktisches Programmieren lernen. Das gilt natürlich auch für Modula. Allerdings hat Modula so viele Elemente aus Pascal geerbt, daß es hier gewagt sei, den Trockenkurs weiter durchzuführen. Heute geht es um Speicherplatz.

Vielleicht besorgen Sie sich eine der auf dem Markt angebotenen Varianten von Modula 2. Die sind nicht mehr allzu teuer. Es gibt sie schon ab 290 DM. Dann können Sie alles viel leichter verstehen und besitzen am Ende eine der flexibelsten Programmiersprachen überhaupt. Übrigens werden Demo-Disketten für PCs und für den Atari angeboten, die um 10 DM kosten. Man kann damit kleinere Programme schreiben und austesten.

Mir ging es bei Pascal immer so, daß ich heimlich dachte, daß man im Grunde auf 64 KByte Speicher bei Turbo-Pascal (Revision < 4.0) begrenzt sei – wenn man große Arrays bauen oder große Datenmengen in einem Schwung behandeln möchte.

Speicherprobleme

Dieser Gedanke ist nicht richtig. Man kann in Pascal größere Speicherräume frei machen und auch so gut wie in einem Stück behandeln. Wie das geht, das sei an einem Beispiel in Modula gezeigt – das vieles über Modula sagt und das im Grunde auch zeigt, wie man es in Pascal macht. Zunächst ist zu sagen, daß bei meiner Version von Modula (zur Zeit M2SDS) jeder Modul sein eigenes Segment besetzt und diese einzelnen Segmente zwar nicht über 64 KByte groß werden dürfen, aber die Komposition von Modulen zu einem Programm schon. Damit ist der statische Speicher innerhalb eines Moduls zunächst auf weniger als 64 KByte beschränkt. Es gibt aber einen Trick, der für Modula wie für Pascal gelingt. Die Sache basiert auf Zeigern und auf dynamischer Speicherverwaltung. Stellen Sie sich vor, Sie haben einen Datentyp ARRAY [0..32767] OF BYTE definiert, indem Sie ihn in der Typendeklaration Paget genannt haben. Bei den meisten Implementationen von Modula auf PCs können sie innerhalb eines Moduls nicht zwei Variablen dieses Typs vereinbaren, denn dann haben Sie schon 64 KByte Speicherplatz fest reserviert, was zu Fehlermeldungen führt, weil dann der Modul-Overhead in dem zur Verfügung stehenden Speichersegment keinen Platz mehr findet. Wenn man sich aber darauf besinnt, daß Pascal und Modula immer den Speicherplatz, der zwischen dem Ende des für das Programm mit seinen Variablen und Konstanten benötigten Speicherraumes und dem Stack-Speicherraum liegt, für den sogenannten Heap zur Verfügung stellt, dann sollte es möglich sein diese vielen KByte (mehrere hundert) für ein Programm nutzbar zu machen. Der Weg ist bei Pascal und Modula fast identisch. Man definiere einen Pointer auf die Array-Struktur.

Beispiel:

```
TYPE
Seite = POINTER TO Paget;
Paget = ARRAY [1..32767] OF BYTE;
.
.
VAR
Page: Seite
```

Ein Pointer wie „Page" ist dazu da, auf Speicherobjekte zu zeigen, die erst während des Programmlaufes bei Bedarf ins Leben gerufen werden (und auch wieder vernichtet werden). Dazu gibt es die Standardprozedur NEW, die auf Zeiger angewendet werden kann:

```
NEW(Page)
```

reserviert hier 32 KByte Speicherplatz auf dem Heap, auf dessen k-te Komponente mit

```
Page ↑ [k]
```

zugegriffen werden kann. NEW(Zeiger) reserviert immer den für ein mit dem Zeiger verbundenen (referenzierten) Datenobjekt benötigten Speicherplatz. Das alles gelingt allerdings nur, sofern der verfügbare freie Speicher auf dem Heap auch noch genügend Platz läßt. In der Praxis sollte man die Seitengröße parametrisieren und mit CONST PageSz = 32767 arbeiten.

Speicherseiten

32 Kbyte sind noch nicht viel. Man definiere einen weiteren Datentyp, nämlich „Seiten":

```
TYP
Seiten = ARRAY [1..10] OF Seite;
```

Und anstelle der Variablen „Page" oben jetzt anders:

```
VAR
Page: Seiten;
```

Jetzt kann man mit

```
FOR i := 1 to 10 DO
NEW(Page[i]);
END;
```

mit einem Schlag 320 KByte reservieren – wenn genügend Platz vorhanden ist. Und mit

```
Page[i] ↑ [k]
```

kann man dann auf den k-ten Platz in der i-ten Seite zugreifen. Diese Technik soll gleich in einem Modul genutzt werden. Zuvor seien noch einige Module beschrieben, die dabei gebraucht werden.

Storage

In jeder Modula-Version gibt es einen Modul in der mitgelieferten Bibliothek, der besonders für Speicher zuständig ist. Er heißt bei mir Storage und enthält Prozeduren, die zum Beispiel auch überprüfen helfen, ob genügend Speicherplatz vorhanden

ist. *Bild 1* zeigt den DEFINITION-Modul der Modulavariante M2SDS. Ehe man NEW benutzt, sollte man mit Storage.Available überprüfen, ob genügend Speicherplatz vorhanden ist. Sie sehen in Bild 1 noch diverse andere Routinen, die teils dazu dienen, den Speicherstand zu messen, teils mithelfen, Speicher zu reservieren.

```
DEFINITION MODULE Storage;
(*------------------------------------------------------------------------
        Copyright (C) 1984 Interface Technologies Corporation.
                          All Rights Reserved.
       Registered users are hereby granted the right to use the
      following program in custom applications.  This permission
        does not include the right to redistribute this code.
------------------------------------------------------------------------*)
(*************************************************************************
 :                                                                       :
 :  Module Storage implements the memory handling routines for           :
 :    SDS.  ALLOCATE and DEALLOCATE are the routines normally            :
 :    used, but lower-level routines are also provided.                  :
 :                                                                       :
 :        Version 1.0                        August 1984                 :
 :        Version 1.1                        June 1985                   :
 *************************************************************************)
  FROM SYSTEM IMPORT ADDRESS;
  EXPORT QUALIFIED ALLOCATE, DEALLOCATE, Available, HeapTop, AllocSeg,
      DeallocSeg, Objptr, Heap;
(*
   ----- Instrumentation -----
*)
  TYPE
    Objptr = POINTER TO
      RECORD
      size : CARDINAL;
      p :
          RECORD
          CASE BOOLEAN OF
            FALSE :
            myself : CARDINAL;
          | TRUE :
            next : CARDINAL;
          END;
          END;
      data : ARRAY [0..1] OF CARDINAL;
      END;
  VAR
    Heap :
      RECORD
              (* ADDRESS of first byte in the heap.          *)
      start : ADDRESS;
              (* ADDRESS of last byte + 1 in the heap.       *)
      end : ADDRESS;
              (* Pointer to next free block to check.       *)
      rover : Objptr;
      END;
  PROCEDURE Available(nbytes : CARDINAL) : BOOLEAN;
(* Return TRUE if nbytes can be allocated from the system heap.       *)
  PROCEDURE HeapTop() : ADDRESS;
(* Return the ADDRESS of the top of the heap                          *)
(************* Allocation from System Heap (used by NEW & DISPOSE) ***)
  PROCEDURE ALLOCATE(VAR ptr : ADDRESS; nbytes : CARDINAL);
(* Allocate a block of storage of at least nbytes and return ptr to it *)
  PROCEDURE DEALLOCATE(VAR ptr : ADDRESS; nbytes : CARDINAL);
(* Deallocate a block of storage that was nbytes long. WARNING: Be sure
   the arguments ptr and nbytes were what got returned from a call to
   ALLOCATE.

*)
(*
   ----- Allocation of Segment Boundaries -----
*)
(* Segment allocation works by allocating more storage than is really
   needed, and bumping the pointer returned to the next paragraph
   boundary. It keeps the original address returned by ALLOCATE in
   the 4 bytes that come before the first byte in the allocated block.
*)
  PROCEDURE AllocSeg(VAR p : ADDRESS; nbytes : CARDINAL);
(* Allocate a block of nbytes on a segment boundary.
   NOTE: Always use DeallocSeg to deallocate these blocks.
         This routine depends upon all allocated blocks having a
         4-byte header with the last two bytes unused.
*)
  PROCEDURE DeallocSeg(p : ADDRESS);
(* Deallocate a block that was allocated by AllocSeg.                 *)
  END Storage.
```

Bild 1. Diese Routinen stellt Storage in der Modulaversion M2SDS zur Verfügung

SYSTEM

Modula sollte den System-Implementator unterstützen. Das geht nicht ohne Rückgriff auf einige physikalische Eigenschaften der unterliegenden Maschine. Man sollte als Programmierer ein Byte reservieren können, ein Maschinenwort, eine Adresse besichtigen und anderes mehr. Allerdings sind das alles sehr maschinenabhängige Objekte, die zum Teil auch von der Implementation abhängen. Niklaus Wirth hat deshalb die Möglichkeit (und Verpflichtung) geschaffen, daß die Implementationsabhängigen Prozeduren und Objekte von einem Modul namens SYSTEM exportiert werden. Das Besondere an diesem Modul ist, daß er in keiner Bibliothek öffentlich zu finden ist, sondern Bestandteil des Compilers ist. SYSTEM wird deshalb als Pseudomodul bezeichnet.

Ein neuer Modul Speicher

In Modula programmieren heißt immer, entweder einen Haupt-Modul zu kreieren oder einen Bibliotheksmodul. Die Bemerkungen oben sollten die Konstruktion eines Bibliotheksmodules mit dem Namen „Speicher" vorbereiten. Dieser Modul soll bei der Bearbeitung von Speicherbereichen (Bildspeicher) nützlich sein. *Bild 2* zeigt sowohl den IMPLEMENTATION MODUL als auch den DEFINITION MODUL, die beide zusammen eine Einheit bilden, aber getrennt von einander (und getrennt von einem Kundenmodul, der sie benutzen will) kompiliert werden können. In Bild 2 finden Sie als Hauptmodul den Modul Test, der Speicher benutzt. Mein Ziel war, möglichst bequem freien Speicher von einem Hauptmodul her behandeln zu können. Dazu schien es mir gut, Speicherseiten zu definieren, die auch von Kundenmodulen benutzt werden können. Deshalb ist im DEFINITION MODULE der Typ Seite definiert, der auf eine Struktur vom Typ Paget zeigt. Paget ist ein varianter Record, weil man mit BYTE in Modula nicht viel anfangen kann. Die Typenprüfung verhindert, daß man Byte-Inhalte ausdrucken kann oder sonstwie gescheit behandeln kann. Die vorliegende Konstruktion legt das Feld der Byte im Speicher genau an die Stelle, an der auch das Feld der CHAR liegt. Das wird später zum Ausgeben der Inhalte genutzt werden. Eine Konstruktion wie der hier, in der im varianten Record das Tagfeld weggelassen ist, kann dazu dienen, die Typenprüfung zu überspielen. Das ist für die systemnahe Programmierung gedacht. Weiter unten ist der Typ Seiten definiert, der ein Feld vom Typ Seite als Struktur festlegt. Die Variable Page reserviert also ein Array von Zeigern, die jeweils auf ein Array von BYTE/CHAR zeigen. Das tun die Page[i] allerdings erst, wenn sie mit NEW initialisiert worden sind.

Nützlich schien mir noch, eine Struktur zu haben, die es erlaubt, konkrete Speicheradressen zu behandeln. Mit derselben

Technik wie oben ist hier der Typ ADDRESS in adr mit zwei CARDINALs überlagert, damit man eine Adresse gegebenenfalls auch betrachten kann. Adresse.off liefert hier den Offset und Adresse.seg liefert die Segmentadresse, wenn Adresse.p vorher einen korrekten Adressenwert zugewiesen bekommen hat. Zum Beispiel könnte man mit

```
Adresse.p:=Page[1]
```

die Adresse der ersten Speicherseite zuweisen.
Umgekehrt wäre zum Beispiel der Bildschirm mit

```
Adresse.off: = 0 und
Adresse.seg: = 11 angesprochen.
```

Eine Freiliste soll konkurrierenden Modulen zeigen, ob noch freier Speicher vorhanden ist. Innerhalb einer Seite soll ein Pegelstand mitnotiert werden, damit bei Zugriffen bekannt ist, wo freier Speicher liegt. In einer Exportliste werden die relevanten Objekte und Prozeduren für Kundenmodule zugänglich gemacht. Im DEFINITION MODUL ist der Kopf der Prozedur Speicheranf(orderung) angegeben, die n Speicherseiten reserviert. Loesche gibt n Speicherseiten frei.

Der IMPLEMENTATION MODUL Speicher

Dieser Modul enthält nun die Prozeduren, die eine Speicheranforderung bearbeiten. Während ein Programmierer, der Speicher benutzen will, unbedingt wissen muß, wie er an die Byte des reservierten Speichers kommt, vielleicht auch wissen muß, wie die Adresse einer Seite von 32 KByte Länge aussieht und wie er die Prozedur aufruft, die ihm n Seiten reserviert, sollte er sich mit dem Wissen, wie die Seiten letzlich programmtechnisch reserviert werden, nicht belasten müssen. Es gehört zum Konzept von Modula, daß so etwas im IMPLEMENTATION MODUL verborgen bleiben soll (der hier natürlich offen vor Ihnen liegt). Wieder werden hier Objekte importiert, die benutzt werden müssen. Available prüft, ob es überhaupt noch Sinn macht,

```
DEFINITION MODULE Speicher;
  FROM SYSTEM IMPORT BYTE, ADDRESS;
  EXPORT QUALIFIED Adresse, Page, Speicheranf, Loesche, Freiliste,
      PegelS;
  CONST
    PageSz = 32767;
    Seitenzahl = 10;
  TYPE
    Seite = POINTER TO Paget;
    Paget =
      RECORD
        CASE INTEGER OF
          1 :
          b : ARRAY [0..PageSz] OF BYTE;
        | 2 :
          c : ARRAY [0..PageSz] OF CHAR;
        END;
      END;
    adr =
      RECORD
        CASE INTEGER OF
          1 :
          p : ADDRESS;
        | 2 :
          off, seg : CARDINAL;
        END;
      END;
    Seiten = ARRAY [1..Seitenzahl] OF Seite;
  VAR
    Adresse : adr;
    Page : Seiten;
    Freiliste : ARRAY [1..Seitenzahl] OF BOOLEAN;
    PegelS : ARRAY [1..Seitenzahl] OF CARDINAL;
  PROCEDURE Speicheranf(n : CARDINAL);
  PROCEDURE Loesche(n : CARDINAL);
  END Speicher.

IMPLEMENTATION MODULE Speicher;
  FROM SYSTEM IMPORT BYTE, ADDRESS;
  FROM Storage IMPORT Available, ALLOCATE, DEALLOCATE;
  VAR
    i, k : CARDINAL;
  PROCEDURE frei32K(VAR Zeiger : Seiten; VAR List : ARRAY OF BOOLEAN;
      Nr : CARDINAL) : BOOLEAN;
    BEGIN
      IF List[Nr]=FALSE THEN
        IF Available(PageSz) THEN
          NEW(Zeiger[Nr]);
          RETURN TRUE;
        ELSE
          RETURN FALSE;
        END;
      ELSE
        RETURN FALSE;
      END;
    END frei32K;
  PROCEDURE Speicheranf(n : CARDINAL);
    BEGIN
      k := 0;
      i := 0;
      WHILE (k<n) AND (i<Seitenzahl) DO
        INC(i);
        IF NOT Freiliste[i] THEN
          INC(k);
          Freiliste[i] := frei32K(Page,Freiliste,i);
        END;
      END;
    END Speicheranf;
  PROCEDURE Loesche(n : CARDINAL);
    BEGIN
      FOR i := 1 TO Seitenzahl DO
        IF Freiliste[i] THEN
          k := i;
        END;
      END;
      WHILE (n>0) AND (k>0) DO
        DISPOSE(Page[k]);
        Freiliste[k] := FALSE;
        DEC(k);
        DEC(n);
      END;
    END Loesche;
  BEGIN
    FOR i := 1 TO Seitenzahl DO
      Freiliste[i] := FALSE;
    END;
  END Speicher.

MODULE Test;
  FROM SYSTEM IMPORT STRING;
  IMPORT Speicher, InOut;
  VAR
    i, j, k, l : CARDINAL;
    Name : STRING[63];
BEGIN
  Speicher.Speicheranf(5);
  FOR i := 1 TO Speicher.Seitenzahl DO
    InOut.WriteCard(i, 6);
    IF Speicher.Freiliste[i] THEN
      Speicher.Adresse.p := Speicher.Page[i];
      InOut.WriteHex(Speicher.Adresse.seg, 6);
      InOut.Write(":");
      InOut.WriteHex(Speicher.Adresse.off, 6);
    END;
    InOut.WriteLn();
  END;
  InOut.WriteString("Bitte Dateinamen eingeben: ");
  InOut.ReadString(Name);
  InOut.WriteLn();
  InOut.OpenInput(Name);
  k := 0;
  i := 1;
  WHILE InOut.Done DO
    InOut.Read(Speicher.Page[i]^.c[k]);
    INC(k);
    IF k>Speicher.PageSz THEN
      k := 0;
      INC(i);
    END;
  END;
  j := 0;
  l := 1;
  WHILE (l<i) OR (j<k) DO
    InOut.Write(Speicher.Page[l]^.c[j]);
    INC(j);
    IF j>Speicher.PageSz THEN
      j := 0;
      INC(l);
    END;
  END;
  InOut.CloseInput();
  InOut.WriteLn();
  Speicher.Loesche(3);
  Speicher.Speicheranf(1);
  FOR i := 1 TO 10 DO
    InOut.WriteCard(i, 6);
    IF Speicher.Freiliste[i] THEN
      InOut.Write("x");
    END;
    InOut.WriteLn();
  END;
END Test.
```

Bild 2. Der Bibliotheksmodul Speicher mit DEFINITION-Teil und IMPLEMENTATION-Teil. Beide werden von Test geprüft

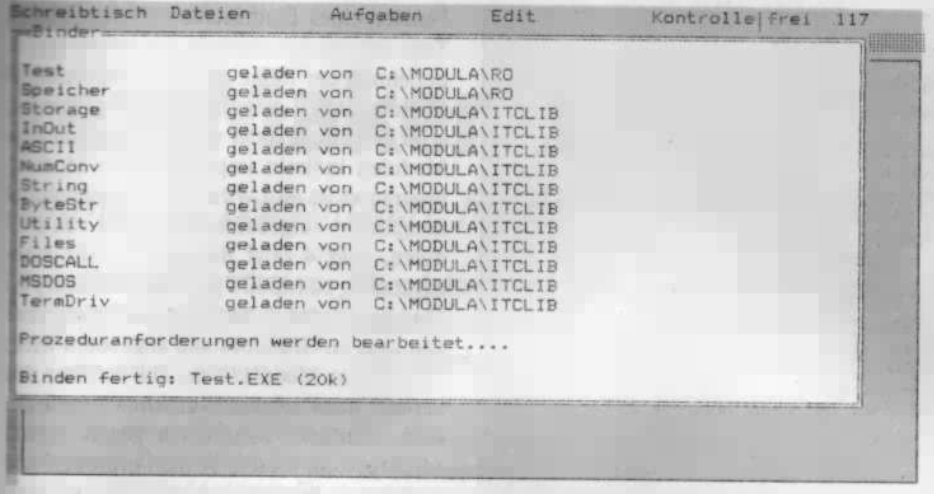


Bild 3. Solch ein Fenster erzeugt mein Modula-Binder. Es sind die Module genannt, die Test direkt oder über andere indirekt importiert

Speicherplatz reservieren zu wollen. ALLOCATE führt eine Reservierung durch. BYTE und ADDRESS müssen wie im DEFINITION MODUL importiert werden. In einem IMPLEMENTATION MODUL müssen nämlich alle zum Bibliotheksmodul gehörenden Objekte in der Importliste genannt werden, auch die schon im DEFINITION MODUL genannten. Umgekehrt gilt das nicht, da sollen sogar möglichst wenig Objekte (nur die, die wirklich zum Definitionsteil beitragen) genannt werden. Die Prozedur Speicheranf berücksichtigt, daß die (hier willkürlich gewählte) Maximalzahl von Seiten 10 beträgt und versucht nach Möglichkeit n Seiten zu reservieren, indem sie die Freiliste prüft und je nachdem die Prozedur frei32K aufruft, die die eigentliche Arbeit macht. Diese Prozedur wiederum übernimmt als Parameter erstens das Array der Zeiger (Page im konkreten Fall), zweitens das Array Freiliste und drittens den Parameter Nr, der die Nummer der zu reservierenden Seite angibt. Wie sie im einzelnen funktioniert, ergibt sich meiner Meinung nach leicht aus dem Listing. Loesche sucht in der Freiliste nach der maximalen Seitennummer und gibt von da rückwärts der Reihe nach n Seiten frei.

Ein Test

Wenn der Modul Speicher in die Bibliothek aufgenommen ist (bei M2SDS durch Editieren und Compilieren der Einheiten in einem „Bibliotheksfenster" des Systems), dann kann man darangehen, einen Hauptmodul zu bauen, der die Sache testet. Mein Test-Modul importiert also den Modul Speicher und dazu noch InOut, einen Modul, den Sie letztes Mal kennengelernt haben. Die beiden Variablen i,k werden zum Zählen benutzt. Im Anweisungsteil von Test wird erst einmal Speicher reserviert. Insgesamt werden 5 × 32 KByte reserviert, also gut 150 KByte. Sodann werden aus Spaß an der Sache die Zahlen 1...10 ausgedruckt und daran jeweils, wenn Speicher.Freiliste[i] TRUE ist, also eine Seite reserviert werden konnte, die Adresse der betreffenden Speicherseite angehängt. Was kann man mit diesem vielen Speicher anfangen? Test soll eine Art Drucker-Spooler sein, der von der Diskette liest und über Pufferspeicher ausdruckt. Dazu sei kurz geschildert, wie man an Dateien herankommt. Mit InOut.OpenInput kommt jetzt eine Eigenschaft des Moduls ins Spiel, auf die Niklaus Wirth großen Wert legt.

Systemunabhängigkeit

Es geht um die Systemunabhängigkeit. Jedes Computersystem besitzt seine Eigenheiten, die besonders dann zu beachten sind, wenn konkrete Maschinen-Ressourcen benutzt werden sollen. Das ist bei Ein- und Ausgaben immer der Fall. Um nun konzeptionell von solchen Eigenheiten absehen zu können, werden von Wirth gewisse Abstraktionen eingeführt. Man denke daran, daß man zum Beispiel die verschiedensten, strukturierten Datenobjekte abspeichern oder auch lesen möchte. Oder daß Massenspeichermedien die Daten blockorientiert haben oder abgeben wollen. In der obersten Abstraktionsebene sollte man sich um solche Details nicht kümmern müssen, man sollte davon abstrahieren können. Nach Wirth ist dort die Struktur, mit der am besten umzugehen ist (und mit der in der Realität auch jeder umgeht) ein Stream. Das ist genau das, was der Name sagt: Ein Strom von Objekten, der entweder dem Modula-Prozessor zum Lesen angeboten wird, oder von diesem in die Peripherie weggeschrieben wird. Das Lesen oder Schreiben geschieht Elementweise. Quelle kann dabei zum Beispiel auch die Tastatur sein, die einen potentiell unendlichen Zeichenstrom erzeugt. Natürlich ist der Bildschirm eine oft benötigte Senke für einen vom Prozessor erzeugten Zeichenstrom. Die Elemente von Streams können Records, Arrays, Wörter .., kurz von jeder sinnvollen Gestalt sein. Der Gedanke von Wirth in Bezug auf Streams ist folgender: Es gibt die geräteunabhängigen Standardstreams, die Textstreams in und out, die im Modulaprozessor per Implementation installiert sind. Ohne besondere Maßnahmen sind sie mit der Konsole (in mit der Tastatur, out mit dem Sichtgerät) verbunden. Es gibt im Modul InOut Prozeduren, die diese Standardstreams abtasten und Zeichen für Zeichen lesen oder schreiben. Dabei passen diese Prozeduren die Zeichen nach Bedarf an. Eine bestimmte (WriteCard) formt zum Beispiel aus einer rechnerinternen Zahl vom Typ CARDINAL eine Ziffernfolge (wobei die erwünschte Länge der Ziffernfolge in einem zweiten Parameter von WriteCard anzugeben ist) für die Interpretation als Dezimalzahl. Im letzten Abschnitt sind die Prozeduren aus InOut aufgezählt worden. Modula-2 ist textorientiert, weshalb der Name der Prozeduren auch ihre Funktion charakterisiert. Die Variable InOut.Done liefert TRUE, wenn eine Operation gelungen ist, sonst FALSE. In den Beispielen früher mußten die Streams in und out nicht erwähnt werden, denn sie waren da stillschweigend mit der Konsole verbunden.

Im Modul Test findet sich die Prozedur InOut.OpenInput angewandt. Sie verbindet Files mit dem Standardstream in. CloseInput löst die Verbindung wieder und in ist dann wieder mit der Konsole verbunden. Nach OpenInput kann von der mit dem Parameter benannten Datei gelesen werden. InOut.Done ist FALSE, falls die Datei nicht geöffnet werden konnte. Im Programm Test kann mit InOut.Write auf die Konsole geschrieben werden, da OpenOutput nicht benutzt wurde. Sie können genau sehen, wie mit Speicher.Page[i]↑.c[k] ein Zeichen angesprochen wird, das k-te Zeichen auf der i-ten Seite. Wenn k eine Seitengrenze zu überschreiten droht, muß i um eins heraufgezählt werden und k wieder auf 0 gesetzt werden, damit eine neue Seite eröffnet und gefüllt wird. Genau so, wie gelesen wird, kann auch mit InOut.Write ein Speicherelement (Page) auf die Konsole geschrieben werden. Zum Abschluß wird in Test Speicher gelöscht und neu reserviert – nur des Testes wegen. Diesmal werden zur Prüfung die reservier-

ASCII - stellt ISO-Standardnamen für ASCII-Kontrollzeichen zur Verfügung.
Utility - stellt Konstanten zur Verfügung, die die Reihenfolge von Bytes in 16- und 32-Bit-Worten belegen.
ByteStr - Routinen zur schnellen Verarbeitung von Byte-Folgen im Speicher.
DOSCALL - Zugang zu den DOS-Funktionen.
Files - Standard-Stream-Ein-/Ausgabeprozeduren unter **M2SDS**. Macht einen stream-orientierten Zugang zum peripheren Speicher des PC (Platten-Dateien), zur Tastatur und zum Bildschirm verfügbar.
Geometry - stellt abstrakte Datentypen und Prozeduren zur Lösung geometrischer Probleme bereit.
InOut - Bietet ein geräte- und systemunabhängiges Hilfsmittel zur Ein- und Ausgabe von Zahlen und Zeichenketten.
MathLib0 - eine Sammlung von Funktionen mit Gleitkomma-Zahlen.
MathLib1 - Gleitkomma-Funktionen mit Benutzung des Coprozessors 8087.
MSDOS - implementiert Streams für Platten-Dateien.
NumConv - Umwandlungsroutinen für CARDINAL, INTEGER, LONGINT in Zeichenketten und umgekehrt.
Program - enthält Prozeduren zur Verwaltung von Overlays.
RealConv - Umwandlung von Gleitkommazahlen in Zeichenketten und umgekehrt.
Screen - direkter Zugang zum PC-Bildschirm: Positionieren der Schreibmarke, Farben, Zugang zu den Graphik-Modi.
ShellVars - Zugriff zu den durch das DOS-Kommando 'SET' erzeugten Umgebungsvariablen und zur Befehlszeile.
Sound - Zugriff zum Tongenerator des PC.
Storage - Routinen zur Reservierung und Freigabe von Speicher im Heap.
String - Routinen zur Handhabung des Typs STRING.
TermDriv - TTY-Treiber für einen Stream-Zugang zur Tastatur und zum Bildschirm des PC.
Terminal - Zugang zu Tastatur und Bildschirm des PC ohne zusätzliche Eingabe-/Ausgabeanforderungen (z.B. auf Platten-Dateien).

Bild 4. Das sind die Standard-Module, die zu meinem Modula-Paket gehören

ten Speicherseiten mit einem x markiert. Das alles bewirkt Test. Sie sehen, daß mit wenig Aufwand ein systemnaher Speicherzugriff über Speicher erreicht werden konnte. Wenn Sie bedenken, daß man den Implementation-Modul Speicher so umbauen könnte, daß er direkt per Adresse Speicherplatz in einer Datei verwaltet, dann könnte man ohne Änderung von Test direkt auf einem externen Massenspeicher arbeiten. Es ist eine der Stärken von Modula, daß man Systeme ökonomisch pflegen kann.

Maschinennähe

Der Modul InOut ist in jedem Modula-System vorhanden, das sich nach Wirth richtet. Er garantiert die Portabilität von Programmen. Der Nachteil ist dabei, daß die Geschwindigkeit nicht optimal ist. Genauer gesagt, das Programm Test liest auf meinem XT-Kompatiblen (4,77 MHz) an 55 Kbyte mehrere Minuten. Das ist der Preis für Abstraktheit. InOut ist gewissermaßen die oberste Schicht einer Hierarchie von Modulen, die immer systemnäher sind, weil sie immer mehr Rücksicht auf die unterliegende Hardware nehmen müssen. In meiner Modula-Variante ist der Modul Files der maschinennähere, der schon Routinen enthält, die blockweise lesen oder schreiben. Allerdings ist er auch schon von MS-DOS geprägt und sicher so nicht auf einem Atari nutzbar. Wieder eine Stufe tiefer findet sich zum Beispiel der Modul DOSCALL, der nun ganz eng auf DOS eingeht. *Bild 3* zeigt das Fenster eines Bindelaufes mit dem Modul Test. Die Modulnamen in der ersten Spalte benennen der Reihe nach die importierten Module. Alles, was nach InOut kommt, wird von diesem oder von ihm importierten Modulen indirekt importiert. Ein Modulaprogramm ist also wirklich eine manchmal kompliziert aufgebaute Hierarchie von Modulen. *Bild 4* zeigt die Module, die standardmäßig zu M2SDS gehören. Es kann von Modulavariante zu Modulavariante schwanken, was an Grundbestand angeboten wird. Der Witz an Modula ist gerade, daß man das System nach seinen Wünschen erweitern kann. Übrigens konnte ich gegen Schluß der Arbeit an diesem Artikel erproben wie gut Modula wartbar und ausbaubar ist. Von A. Meier aus Bonstetten, der mich bei der Arbeit sehr durch gute Ratschläge unterstützt hat, erhielt ich eine Diskette, auf der die Implementation-Module von InOut und Files in neuer Form gegeben waren. Ich konnte sie einfach in mein System übernehmen, compilieren, ohne die zugehörigen Definition-Module überhaupt nur anschauen zu müssen und dann den Modul-Test neu binden. Das Ergebnis war eine rapide Verbesserung der Lesegeschwindigkeit von der Platte. Wohlgemerkt, nur durch Austausch von Modulen, die irgendjemand irgendwo (mit viel Intelligenz) und ohne meine Programme zu kennen, programmiert hat.

Spruch des Monats

„Viele Entwicklungen, die wir machen, sind viel zu gefährlich für die Menschheit in ihrem heutigen Zustand. Nicht wenige nehmen an, es lasse sich eine Maschine bauen, mit der man im Grunde alles in der Welt erfassen kann. Der Traum vom Wundercomputer, der einmal das Erbe der vergänglichen Menschen antreten solle, ist Zauberglauben. Man muß auch einmal bereit sein nein zu sagen."

Prof. Joseph Weizenbaum vom MIT bei einer Tagung der evangelischen Akademie Tutzing, gefunden in „Kirche und Medien", Heft 27.

Lore Schönrock

TextMaker 1.0

SoftMakers Textsoftware für den Low-Cost-Bereich

Wordstar wird immer noch häufig eingesetzt, und immer wieder graben es Software-Häuser aus, um mit wenigen Änderungen eine Textsoftware für den Low-Cost-Bereich zu produzieren. TextMaker 1.0 ist ein solches Produkt aus dem Hause SoftMaker in Nürnberg.

Der Versuch, WordStar zu überarbeiten, einige Neuerungen beizubringen und damit den geänderten Bedürfnissen anpassen zu wollen, kann nicht gerade als zeitgemäß bezeichnet werden. Doch bietet TextMaker ausreichend Leistung für den Heim- und Hobbybedarf.

TextMaker macht nach dem Aufruf eines Dokumentes den Bildschirm zum Schreiben frei, nur die oberen drei Zeilen bieten Menüleiste, Format- und darunter Statuszeile. Die Formatzeile ist wie in WordStar mit einem Tabulatorraster versehen. In die Menüleiste führt F10 und zurück zum Text ESCAPE. Bleiben wir bei obigem Beispiel und verschieben einen Textblock. In der Menüleiste muß dazu BLOCK gewählt werden – am schnellsten mit dem Anfangsbuchstaben. Es öffnet sich ein Pull-Down-Menü, der Blockanfang ist invertiert angezeigt und braucht nur noch bestätigt zu werden. Der Cursor steht dann im Text und muß zum Blockende bewegt werden. Dazu können die Pfeiltasten oder – wie in WordStar – CTRL in Verbindung mit dem entsprechenden Buchstaben mehrmals gedrückt werden. Ist die Position erreicht, muß das Menü, dann die Option BLOCK und im geöffneten Menü Blockende gewählt und bestätigt werden. Wer WordStar kennt, wird wahrscheinlich bald auf CTRL-KB, CTRL-KK und die Blockbearbeitungsfunktionen zurückgreifen. Beim Löschen ist der Weg über das Menü zu umständlich. Schneller ist ein Textblock mit CTRL-KY und ein einzelnes Zeichen mit der Korrekturtaste gelöscht, als das entsprechende Pull-Down-Menü aufgerufen.

Die Formatierungsmöglichkeiten sind bescheiden, aber für den Anwenderkreis durchaus ausreichend. Einrückungen – wie sie bei längeren Texten oder für Unterpunkte gängig sind – werden mit einer über die Textpassage einzufügenden Formatzeile vorgegeben. Diese Zeile hat mit dem Nummernzeichen zu beginnen, damit sie nicht im Ausdruck erscheint, und mit Leerschritten sowie einem L (linker Rand) und R (rechter Rand) zu markieren. Müssen verschiedene Einrückungen gesetzt werden, wird die Merkfähigkeit des Anwenders arg strapaziert. Umgehen kann man das Ganze wohl nur durch Kopieren der benötigten Formatzeile. Beim Bearbeiten des Textes muß darüber hinaus wie bei WordStar der Text auch noch mit dem Formatierbefehl CTRL-B ins Format gebracht werden. Dabei sollte man wenigstens gleich im rechten Randbereich trennen.

Programm-Merkmale auf einen Blick

Nutzt WordStar-kompatiblen Befehlssatz
Erzeugt ASCII-Dateien
Zeigt Miniatur-Layout mehrerer Dokumentenseiten am Bildschirm
Öffnet bis zu neun Fenster
Phonetisches Suchen
Stichwort- und Inhaltsverzeichnisse
Fußnoten

Wie der Text auf das Papier zu schreiben ist, wieviel Platz von der Papierkante oben, unten und an den Seiten frei bleiben soll, wird unter OPTIONEN abgefragt. Hier ist auch gleich der Abstand zur Kopf- und Fußzeile vorzuschreiben. Alle weiteren Wünsche sind mit Punktbefehlen im Dokument zu vermerken. So beispielsweise, wenn an einer bestimmte Stelle ein Seitenumbruch vorgenommen werden muß oder stellenweise ein anderer Zeilenabstand benötigt wird.

Punktbefehle werden neben den Formatmerkmalen für die Kopf- und Fußzeile, die Seitennumerierung und auch für den Mischdruck von Serienbriefen und Textbausteinen benötigt. Es sind auch hier die gleichen wie bei WordStar:

.pa für den Seitenumbruch, .he für die Kopfzeile, .fo für die Fußzeile, .fi für den Textbaustein.

SoftMaker wählt ein ungewöhnliches und doch praxisbezogenes Beispiel in der Bedienungsanleitung, um zu zeigen, daß sich die Punktbefehle für den Mischdruck auch zum Ausstellen von Rechnungen nutzen lassen. Denn das Programm kennt die vier Grundrechenarten. Zum Nachmachen empfehlen wir es jedoch nicht, denn die Zahlenkolonne wird nach der ersten Zahl ausgerichtet und nicht nach den vollen Markbeträgen. Schneller als die Zahlen in die Punktbefehle eingetragen werden, ist die Rechnung komplett mit dem Taschenrechner gemacht. Das Adreßprogramm ist für den Anwenderkreis der Software weitaus interessanter. Denn hier können die Adressen von Bekannten- und Freundeskreis aufgenommen und für Briefe genutzt werden. Eine Adreßdatei besteht laut Angabe im Bedienungsbuch aus maximal 100 Einträgen. Das wird gewiß für diesen Zweck reichen. Die Adreßaufnahme erfolgt über ein Eingaberaster, dessen Feldbezeichnungen und Feldlängen sich nicht ändern lassen. Dadurch läßt sich die Videothek oder Bibliothek daheim nicht sehr komfortabel erfassen.

Ein Hilfsmittel gegen Datenverlust ist das Zwischenspeichern. TextMaker nimmt es alle acht Minuten vor. Der Takt läßt sich auch ändern. Diese Funktion ist für das Bearbeiten umfangreicher Texte sinnvoll, solange man sich von dem begleitenden Piepser nicht irritieren läßt.

Auch Stichwort- und Inhaltsverzeichnis sind Funktionen, die für umfangreiche Texte benötigt werden. Bei TextMaker wirken sie eher rudimentär. Mit den Punktbefehlen kann man eine Markierung am Beginn der Zeilen setzen, die ins Inhaltsverzeichnis übernommen werden sollen. Für den Index muß es jeweils eine Anfangs- und Endmarkierung sein. Die Markierungen geben den Befehl, eine Diskettendatei einzurichten und alle Einträge mit den entsprechenden Seitennummern versehen auszudrucken. Sie werden im Text mitgeführt.

Bemerkenswert ist die Möglichkeit, alle Buchstabentasten mit Floskeln oder Befehlsketten von maximal 70 Zeichen zu belegen. „Mit freundlichen Grüßen" bräuchte nur ein einziges Mal geschrieben zu werden und wäre mit ALT in Verbindung mit dem zuständigen Buchstabe jederzeit abrufbereit. Für die private Korrespondenz ist damit sicherlich eine Erleichterung verbunden. Textmaker kostet 148 DM. □

Günther Sternberg

Overlays unter Turbo 4.0

Mit Turbo-Pascal 4.0 ist nicht nur die 64-KByte-Grenze gefallen, sondern auch, zumindest in den Augen von Borland, die Notwendigkeit Overlays zu verwenden. Einige Aufgaben lassen sich jedoch ohne Overlays nicht realisieren. Ein Programmpaket versucht diese Lücke zu schließen.

Als professioneller Anwender konnte ich es kaum glauben, daß Borland in der neuesten Version seines ansonsten hervorragenden Turbo-Pascal-Compilers auf Overlays verzichtet. Bei einem Gespräch auf der Systems 87 mit der Firma Heimsoeth, dem deutschen Distributor von Turbo-Pascal, erhielt ich die Auskunft, daß mit dem Wegfall der 64-KByte-Grenze für den Programmcode Overlays nicht mehr nötig seien. Dieser Aussage liegt sicherlich eine Fehleinschätzung über Nutzen und Anwendung von Overlays zugrunde. So ermöglicht der Einsatz von Overlays die Schaffung von Programmen mit mehr als 640-KByte Code. Meist ist es aber der Wunsch möglichst viel freien Speicher für Daten zur Verfügung zu haben, der einem zur Aufteilung eines Programms in mehrere Overlays treibt. Eine Reihe von Programmen, wie Lotus oder Open Access, wären ohne diese Programmiertechnik undenkbar.

Der Ausweg

Die amerikanische Software-Schmiede Turbo-Power-Software hilft Anwendern der neuesten Turbo-Pascal-Version mit ihrem Overlay-Manager 4.0 weiter. Der Overlay-Manager besteht aus einer 5¼-Zoll-Diskette und einem knapp 30seitigen englischen oder deutschen Handbuch. Zum Betrieb des Programms benötigt man lediglich einen IBM-PC/XT/AT-kompatiblen Rechner mit DOS Version 2.x oder 3.x und natürlich Turbo-Pascal Version 4.0. Wer die auf der Diskette enthaltenen Quellen abändern möchte, der benötigt zusätzlich den Makro-Assembler (MASM) von Microsoft (Version 4.0 oder höher). Wohlgemerkt, der reine Anwender des Overlay-Managers benötigt keinen Assembler. Alle nötigen Programmteile liegen in ablauffähiger Form vor.

Leistungsfähiger als die Overlay-Verwaltung unter Turbo 3.0

Die Funktionsweise des Overlay-Managers läßt sich am besten durch einen Vergleich mit der Overlay-Funktion von Turbo-Pascal 2.x oder 3.x beschreiben. An dieser Stelle wird davon ausgegangen, daß Sie ein wenig vertraut sind mit der Handhabung von Overlays in bisherigen Versionen von Turbo-Pascal. Bisher war eine Funktion oder eine Prozedur die kleinstmögliche Einheit für ein Overlay. Alle unmittelbar nacheinander deklarierten Overlay-Funktionen oder Prozeduren gehörten zu einer Overlay-Gruppe und wurden gemeinsam in einer Overlay-Datei (z. B. TEST.000) gespeichert. Beim Laden des Programms reservierte Turbo-Pascal für jede Overlay-Gruppe einen Bereich im Hauptspeicher. Die Größe dieses Speicherbereichs entsprach dem Platzbedarf der größten Routine der entsprechenden Overlay-Gruppe. Da immer nur eine Routine einer Overlay-Gruppe in diesen Bereich geladen wurde und bis zur vollständigen Abarbeitung in diesem Bereich verblieb, war ein gegenseitiger Aufruf der Routinen einer Overlay-Gruppe nicht zulässig.

Beim Overlay-Manager bilden nicht die einzelnen Funktionen oder Prozeduren ein Overlay, sondern eine Unit. Da eine Unit in der Praxis aber aus nur einer einzigen Funktion oder Prozedur bestehen kann, ist das im Endeffekt eine reine Formsache. Die in einem Programm benötigten Overlay-Units lassen sich zu einer oder mehreren Overlay-Gruppen zusammenfassen. Für jede Overlay-Gruppe wird im Hauptspeicher ein bestimmter Bereich reserviert. Im Gegensatz zur Overlay-Verwaltung von Turbo-Pascal können sich die Units des gleichen Overlay-Bereichs gegenseitig aufrufen. Angenommen zu einer Overlay-Gruppe gehören die Units A und B und innerhalb der Unit A wird eine Prozedur aus der Unit B aufgerufen. Sobald der Aufruf der Unit B erfolgt, speichert der Overlay-Manager die aktuelle Position und den aktuellen Zustand in Unit A ab und lädt die Unit B in den gleichen Speicherbereich. Nach der Abarbeitung der Prozedur in der Unit B wird die ursprüngliche Unit A wieder geladen und die Programmausführung an der alten Stelle fortgesetzt. Sollten Sie sich angesichts dieser Möglichkeit fragen, warum man dann überhaupt noch mehrere Overlay-Gruppen verwenden soll, hierzu folgende Erläuterung. Der gegenseitige Aufruf zweier Units einer Overlay-Gruppe kostet viel Zeit für das Nachladen der Overlays. Befinden sich beide Units in verschiedenen Overlay-Gruppen, werden beide Units nur einmal geladen, da jede Overlay-Gruppe über ihren eigenen reservierten Speicherbereich verfügt.

Grenzen des Overlay-Managers

Die auf der Diskette enthaltene Standard-Version des Overlay-Managers läßt die Bildung von maximal 6 Overlay-Gruppen mit jeweils maximal 51 Overlay-Units zu. Die Summe aller Units (inklusive der Nicht-Overlay-Units) darf 255 nicht überschreiten. Nach den Erfahrungen des Herstellers sind die Grenzwerte für die meisten Anwendungen völlig ausreichend. Der Overlay-Manager belegt mit diesen Einstellungen ungefähr 2 KByte Code und 1 KByte Daten. Auf Kosten eines größeren Speicherbedarfs läßt sich die Anzahl der Overlay-Gruppen und die Anzahl der Units pro Gruppe auf jeweils maximal 255 erhöhen. Die Anzahl der Units insgesamt wird lediglich durch den verfügbaren Speicherplatz begrenzt. Zur Änderung eines Parameters ist in der Quelle eine Konstante auf einen neuen Wert zu setzen und das Programmpaket neu zu compilieren. Hierzu benötigt man dann allerdings einen Makro-Assembler.

Speichererweiterungskarten werden unterstützt

Das Nachladen der einzelnen Overlay-Units von Diskette oder Festplatte ist ein relativ langsamer Vorgang. Einen erheblichen Zeitgewinn erzielt man beim Laden der Overlays von einer RAM-Floppy oder einer Speichererweiterungskarte (Expanded Memory). Der Anwender kann, auch nachträglich, mit einem Installationsprogramm den Overlay-Manager zur Ausnutzung einer Speichererweiterung gemäß der Lotus/Intel/Microsoft-Norm veranlassen. Der Over-

lay-Manager lädt dann bei Programmstart den Code der Overlays zunächst in das Expanded Memory und holt während der Programmausführung bei Bedarf die Overlays sehr schnell aus diesem Speicherbereich.
Ein kleiner Hinweis für Benutzer, die Ihre Speichererweiterungskarte als RAM-Floppy benutzen. Der Overlay-Manager verwendet mit der standardmäßigen Einstellung automatisch immer eine vorhandene Erweiterungskarte. Bei der Nutzung des Zusatzspeichers mit einem RAM-Disk-Treiber kommen sich der Overlay-Manager und der Treiber gegenseitig in die Quere und ein Systemabsturz ist meistens die Folge. In diesem Falle sollte im Overlay-Manager mit dem Installationsprogramm die Nutzung des Zusatzspeichers gesperrt werden.

Overlay-Manager in der Praxis

Während der Compilierung eines Programms auf Diskette (Erzeugung der EXE-Datei) kümmert sich Turbo-Pascal nicht um den Speicherbedarf. Es lassen sich daher Programme übersetzen, die weit mehr als 640 KByte Speicher benötigen. Der Overlay-Manager zerlegt nachträglich die so gewonnene EXE-Datei in eine neue EXE-Datei und eine Datei mit den gesamten Overlays. Zum Erstellen eines Programms mit Overlays sind folgende Schritte nötig:
In der Uses-Anweisung des Hauptprogramms muß man zuerst eine Unit mit dem Namen OVERMGR angeben. Weiterhin ist der Compiler mit der Option {$T+} zur Erzeugung einer TPM-Datei zu veranlassen. Als nächstes wird das Programm compiliert (Erzeugung der EXE-Datei) und die erhaltene TPM-Datei mit dem Turbo-Programm TPMAP in eine MAP-Datei umgewandelt. Nun muß man in einer Textdatei mit dem Zusatz .OSP nur noch alle Overlay-Units namentlich unter Angabe der Overlay-Gruppennummer (0 bis 5) aufführen. Die Datei könnte beispielsweise folgendermaßen aussehen:

```
Lade_Daten 0
Drucke_Daten 1
Berechne 0
Ausgabe 1
```

In diesem Beispiel würden die Units „Lade_Daten" und „Berechne" zur Overlay-Gruppe 0 und die beiden restlichen Units zur Overlay-Gruppe 1 gehören. Im letzten Schritt wird das Programm OVERLAY aufgerufen, daß die Zerlegung der ursprünglichen EXE-Datei durchführt. Die neue EXE-Datei und damit das Programm lassen sich dann sofort aufrufen.
Unter Verwendung der Kommandozeilen-Version des Turbo-Pascal-Compilers lassen sich alle Schritte bequem mit einem Batch-Job erledigen.

Nur geringe Einschränkungen

Eine angenehme Eigenschaft dieses Programmpakets liegt in dem Umstand, daß es nahezu keine Einschränkungen gibt. Einige Units von Turbo-Pascal (z. B. Unit SYSTEM) und eigene Units mit Interrupt-Routinen müssen ständig im Speicher bleiben und dürfen daher nicht als Overlay deklariert werden. Weiterhin gibt es bei der Übergabe eines Strings als formalen Parameter eine Einschränkung zu beachten. Das Handbuch informiert auf 4 Seiten ausführlich über die zu beachtenden Dinge.

Chain-Befehl als Zugabe

In Turbo-Pascal 4.0 ist außer der Overlay-Verwaltung auch der Chain-Befehl weggefallen. Der Overlay-Manager enthält Routinen zur Emulation dieses Befehls. Weiterhin sind eine Reihe von Funktionen und Prozeduren zur Übergabe von Daten an das mit dem Chain-Befehl aufgerufene Programm enthalten.

Overlays sind nicht tot

Der knapp über 100 DM teuere Overlay-Manager 4.0, im Vertrieb von H+B EDV in Tettnang und Enz EDV in Bad Homburg, ist eine sinnvolle Ergänzung von Turbo-Pascal für alle Programmierer, die auf Overlays nicht verzichten können oder wollen.
Inzwischen scheint sich auch bei Borland ein Sinneswandel vollzogen zu haben. Für zukünftige Versionen von Turbo-Pascal soll ein „intelligenter" Overlay-Manager in Bearbeitung sein. Worin die Intelligenz dieses Overlay-Managers liegen soll, ist noch ein Rätsel. Aber Borland ist ja immer für eine Überraschung gut.

Wer hat wann wo was publiziert?

Auf die Verwaltung von Literaturstellen spezialisiert ist das Programm Microlit. Zu einer Publikation gehören unverzichtbar die Angaben über Autoren, Titel, Verlag und bei Zeitschriften Bandnummer, Jahrgang und Seitennummern. Microlit läßt sogar zehn Autoren und fünf Schlüsselworte pro Publikation zu, für die in den Masken bereits Platz gelassen wurde. Die hohe Zahl von Autoren erklärt sich wohl dadurch, daß das Programm in Zusammenarbeit mit einem Mediziner entstand und in diesem Anwendungsbereich anscheinend solch große Autorengruppen durchaus nicht unrealistisch sind. Für andere Bereiche jedoch hätte man sich eine Verminderung der Autorenzahlen zugunsten von mehr Schlüsselwörtern gewünscht.
Microlit verwaltet mit, ob es eine Originalarbeit des Benutzers ist, an welchem Ablageort sich die Publikation befindet und um welche Literaturart es sich handelt. Es werden zunächst fünf Literaturarten unterschieden: Abstract, Buch, Journal, Buchartikel und Zeitung; der Benutzer kann aber bis zu zehn Literaturarten vereinbaren (jede hat eine einbuchstabige Abkürzung).
Für jede Literaturart lassen sich in Format-Definitionen eigene Masken festlegen, die u.a. über Trennungszeichen, Schriftarten und Reihenfolge der Felder beim Ausdruck entscheiden. Die Gesamtheit der erfaßten Publikationen ist durchnumeriert, jede erfaßte Publikation hat also ihre eigene Publikationsnummer.
Die Maske zur reinen Verwaltung (Blättern, Neueingabe, Ändern usw.) zeigt zusätzlich am oberen Bildschirmrand ständig die Gesamtzahl der Publikationen und das Datum an. Die Uhrzeit, die sich im Handbuch unter dem Datum befindet, war in der Testversion von Microlit an dieser Stelle (zumindest nicht standardmäßig) vorhanden.
Jede Publikation läßt sich in einem maximal 4000 Buchstaben langen Abstract inhaltlich kurz charakterisieren.
Microlit ist gut ausgestattet mit Hilfestellungen, die über die Funktionstaste F1 erreichbar sind und abhängig von der Aufrufstelle die passenden Informationen bieten.
Negativ fiel auf, daß bei einem Wechsel zwischen Hilfe und Eingabemaske die Bildschirminhalte sich nicht immer wechselweise ausblenden, sondern manchmal überlagern.
Bei der Suche nach bestimmten Schlüsselwörtern können mit „Teilsuche" Suchbegriffe auch dann gefunden werden, wenn sie innerhalb eines anderen Schlüsselwortes vorkommen; das entspricht in der Funktion den bekannten Wildcards bzw. Jokern.

Dieter Strauß

Moderner PC-Assembler

MASM 5.0 zeigt, was er kann

Mit PC-Assemblern ist Microsoft schon lange auf dem Markt vertreten. Seit kurzem ist die neueste Version des PC-Assemblers – MASM 5.0 – erhältlich. Auffällig sind die vereinfachten Segment-Direktiven, die Anfängern den Einstieg erleichtern und Profis das Programmieren angenehmer machen.

Das erste, was an dem MASM 5.0 auffällt, sind die drei Handbücher. Allein dem eigentlichen Assembler ist ein 468seitiges Handbuch gewidmet. Das Handbuch zum Debugger Codeview und den Utilities ist 400 Seiten dick und die Anleitung zur Mischung von Programm-Modulen mehrerer Sprachen ist immerhin noch 140 Seiten stark. Allein in diesem Punkt unterscheidet sich MASM 5.0 ganz deutlich von seinen Vorgängern. Leider sind die Handbücher alle in Englisch und eine Übersetzung ist derzeit nicht geplant.

Die wichtigsten Neuerungen sind:

- Spezielle Befehle des 80386-Prozessors werden unterstützt
- Ein symbolischer Debugger in Fenstertechnik, Codeview (*Bild 1*) genannt, löst den bei früheren Versionen mitgelieferten Symdeb ab
- Vereinfachte Segment-Direktiven
- Vereinfachte Konvention für Unterprogramme

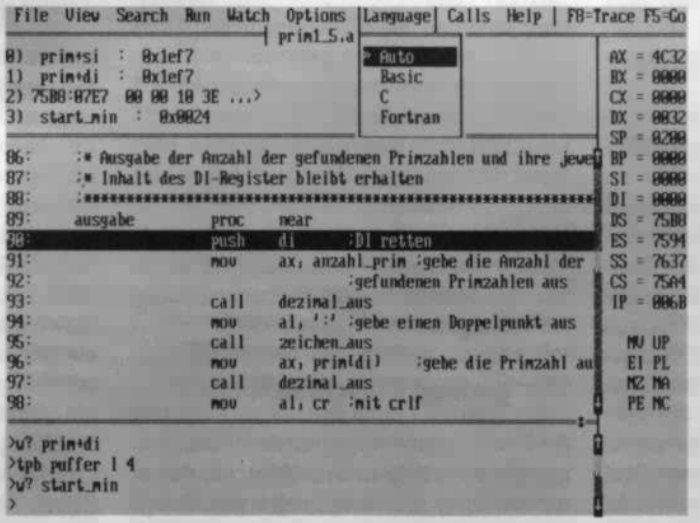


Bild 1. Der symbolische Debugger Codeview verwendet ausgiebig die Fenstertechnik

Bei zahlreichen Kontakten mit Anwendern ist mir aufgefallen, daß sich besonders Assembler-Neulinge schwer tun, mit den Segment-Direktiven früherer MASM-Versionen umzugehen. Offenbar hat Microsoft das erkannt und vereinfachte Segment-Direktiven entwickelt. Es ist jedoch für den erfahrenen Anwender wichtig zu wissen, daß er weiterhin die gewohnten Direktiven verwenden darf.

Früher lauteten z. B die Anweisungen für das Stack-Segment wie folgt:

```
STACK SEGMENT PARA STACK 'STACK'
db 256 dup (?)
STACK ends
```

Bei dieser Konvention steht der Name des Segments vor dem Schlüsselwort SEGMENT. Mit PARA legt der Programmierer fest, daß das Stack-Segment auf einer Paragraphgrenze (Adresse durch 16 teilbar) beginnen soll. Nach PARA folgt die Angabe zur Kombinierung (Combine) von Segmenten. Der Klassenname (Class) ist hier ebenfalls STACK. Die Bedeutung, der auf den ersten Blick unsinnig erscheinenden Segment-Deklaration, wird erst klar, wenn man Programme schreibt, die aus mehreren Segmenten bestehen. Linkt man Objektmodule (OBJ-Dateien), die von Compilern erzeugt wurden, mit Objektmodulen aus Assembler-Programmen, muß man sich genau an die jeweiligen Segment-Direktiven halten. Da die in mc veröffentlichten Assembler-Programme sehr kurz sind und jeweils nur aus einem Objektmodul bestehen, veröffentlicht mc die Programme mit kürzeren Segment-Direktiven. Der Assembler verwendet dann die voreingestellten Direktiven.

MASM 5.0 kürzt den Wildwuchs der Segment-Direktiven radikal. Die obige Anweisung sieht so aus:

```
.STACK 256
```

Mit der neuen Anweisung wird das Leben sowohl eines Profis als auch des gelegentlichen Programmierers leichter.

Auch die Segment-Direktiven für das Code- und das Daten-Segment sind einfach. Sie lauten .CODE für Code-Segmente und .DATA für Daten-Segmente.

Wie bereits bei C-Compilern üblich, muß man das Speichermodell explizit angeben. Wenn der Code und die Daten in je ein Segment mit 64 KByte passen, gibt man, um das Speichermodell zu definieren, .MODEL SMALL ein. Wichtig ist der Model-Befehl, wenn aus mehreren Programm-Modulen, worunter auch Module aus anderen Sprachen von Microsoft gehören, ein Programm erstellt werden soll. Bei COM-Programmen darf man allerdings die vereinfachten Segmentanweisungen nicht verwenden. Die Segment-Anweisungen eines COM-Programms sind jedoch so einfach, daß man auf die Model-Anweisung TINY leicht verzichten kann.

Die DOSSEG-Direktive ordnet bei Programmen, die aus mehreren Segmenten bestehen, die einzelnen Segmente entsprechend der Microsoft-Konvention.

Mit der zunehmenden Verbreitung von 80386-Maschinen wird von einem Assembler natürlich auch deren Unterstützung verlangt. MASM 5.0 kennt die neuen 80386-Befehle und die Befehle des Numerik-Prozessors 80387.

Auch für die Anwender, die bisher mit anderen Assemblern arbeiteten, bietet MASM 5.0 neues. Unterprogramme mußte man in früheren MASM-Versionen wie folgt „einrahmen“:

label PROC (NEAR oder FAR)
Anweisungen
RET (Konstante)
label ENDP

Die kursiv gedruckten Bezeichnungen muß man in einem Programm mit dem jeweiligen Label bzw. den jeweiligen Anweisungen ersetzen. Hier steht *label* für den Namen des Unterprogramms. Ausdrücke in Klammern sind optional. Bei Unterpro-

grammen innerhalb des gleichen Segments kann die Direktive NEAR entfallen.
Wem die herkömmliche Schreibweise zu aufwendig oder nicht vertraut ist, darf bei Unterprogrammen, die innerhalb des gleichen Code-Segments stehen, folgendes eingeben:
label
Anweisungen
RETN (Konstante)
Bei Unterprogrammen außerhalb des gleichen Codesegments sieht das so aus:
label LABEL FAR
Anweisungen
RETF (Konstante)
mc wird weiterhin die herkömmliche Schreibweise verwenden, da sie Unterprogramme deutlicher hervorhebt.
Um die Unterschiede zwischen MASM 4.0 und MASM 5.0 klar darzustellen, habe ich ein Demo-Programm geschrieben, das wie der DOS-Befehl Time die Systemzeit ausgibt. Im Gegensatz zum DOS-Befehl kann man jedoch die Systemzeit nicht einstellen. In *Bild 2* ist die MASM-4.0-Version und in *Bild 3* die MASM-5.0-Version dargestellt.
Damit keine Mißverständnisse entstehen: Selbstverständlich kann MASM 5.0 auch das konventionell geschriebene Programm korrekt übersetzen. Während der Programmentwicklung fiel mir auf, daß MASM 5.0 Fehler entdeckt, die MASM 4.0 durchgehen läßt. So stieg bei einer Vorversion des Programms erst der Linker mit einer abstrusen Fehlermeldung aus. MASM 5.0 bemerkte den Fehler sofort.
Um die Zukunftssicherheit des Assemblers zu testen, starteten wir ihn in der Kompatibilitätsbox von OS/2. Wie bei seinem Vorgänger kann man in der Kompatibilitätsbox Assembler-Programme schreiben, die im geschützten Modus laufen sollen. Zum Linken der Objektmodule muß man den Linker/2 verwenden, der mit OS/2 geliefert wird. Leider ist zum Thema OS/2 in den Handbüchern nichts zu finden.

```
cr              equ     0dh                     ;Carriage Return
lf              equ     0ah                     ;Line Feed
ascii_0         equ     '0'                     ;Codierung der Zahl 0 in ASCII
stdaus          equ     1                       ;Handle für Standard-Ausgabe
;
_DATA           SEGMENT WORD PUBLIC 'DATA'                  ;Daten-Segment
string          db      cr,lf,'Uhrzeit: 00:00:00.00',cr,lf  ;Systemzeit
;  Zehnerstelle:        0  0   0000000011111111122  Position der einzelnen
;  Einserstelle:        0  1   2345678901234567890  Byte im String
string_laenge   equ     $-string                ;$=Adreßzähler
_DATA           ends
;
STACK           SEGMENT PARA STACK 'STACK'      ;Stack-Segment
                db      256 dup (?)             ;Stack: 256 Byte
STACK           ends
;
_TEXT           SEGMENT WORD PUBLIC 'CODE'      ;Code-Segment
                assume cs:_TEXT, ds:_DATA, ss:STACK   ;Segment-Zuordnung
main            proc    far                     ;Hauptprogramm
start:          mov     ax, _DATA               ;ds-Register laden
                mov     ds, ax                  ;über ax
                mov     ah, 2ch                 ;Systemzeit abfragen
                int     21h                     ;ch:h, cl:m, dh:s, dl:h_s
                mov     string+12, ch           ;Speichern der Systemzeit
                mov     string+15, cl           ;im Feld string als
                mov     string+18, dh           ;Dualzahl
                mov     string+21, dl           ;
                mov     di, offset string       ;Startadresse des Feldes in di
                mov     bx, 21                  ;21 Zeichen ohne letztes crlf
schleife1:      mov     al, [bx+di]             ;bx+di --> auf ein Byte des Strings
                call    dual_ascii              ;Umwandeln Binär -> ASCII
                sub     bx, 3                   ;bx := bx - 3
                cmp     bx, 10                  ;Vergleich mit dem vorletzten
                                                ;Element des Ausgabebereichs
                jnb     schleife1               ;Abbruchbedingung erfüllt?
                call    ausgabe                 ;wenn ja: Erfolgsmeldung
                mov     ax, 4c00h               ;Rückkehr ins DOS
                int     21h                     ;
main            endp                            ;Ende des Hauptprogramms
;
ausgabe         proc    near                    ;Unterprogramm
                mov     bx, stdaus              ;Handle für Standard-Ausgabe
                mov     cx, string_laenge       ;Länge der Meldung in cx
                mov     dx, offset string       ;Offset der Meldung in dx
                mov     ah, 40h                 ;DOS-Funktion 40h
                int     21h
                ret                             ;Rückkehr ins Hauptprogramm
ausgabe         endp                            ;Ende des Unterprogramms
;
dual_ascii      proc    near                    ;umzuwandelndes Zeichen in al
                push    bx                      ;bx sichern
                xor     ah, ah                  ;ah := 0
                mov     cx, 10                  ;cx := 10
schleife2:      xor     dx, dx                  ;dx := 0
                div     cx                      ;dl := (dx:ax)/10
                add     dl, ascii_0             ;ASCII := Dual + '0'
                mov     [bx+di], dl             ;ASCII-Ziffer im String speichern
                dec     bx                      ;nächste Ziffer
                cmp     ax, 0                   ;Dualzahl konvertiert?
                jnz     schleife2
                pop     bx                      ;bx restaurieren
                ret                             ;Rückkehr in aufrufende
dual_ascii      endp                            ;Programm
_TEXT           ends                            ;Ende des Code-Segments
end             start                           ;Programm-Einsprungpunkt bei start
```

Bild 2. Ein Demo-Programm mit konventionellen Assembler-Direktiven

Zahlreiche Optionen

Wie seine Vorgänger kann man MASM 5.0 mit einigen Optionen starten. In der *Tabelle* sind alle Optionen aufgeführt.
Interessant ist die Option /W zum Einstellen der Schwelle für Fehlermeldungen. Um diese Option verständlich erscheinen zu lassen, gebe ich einen kurzen Abriß zur Geschichte des MASM.
Beim Übergang von der MASM-Version 3.0 auf 4.0 hatte sich wenig geändert. Lediglich die strengere Typenprüfung von Variablen und die Beseitigung einiger Fehler zeichneten die Version 4.0 aus. Besonders die strengere Typenprüfung bereitete einigen Anwendern Kummer. Viele lässig programmierten älteren Assembler-Programme ließen sich mit der neuen Version nicht mehr fehlerfrei assemblieren.
Um das Problem zu lösen, werden alle Typenfehler beim MASM 5.0 nur als Fehlerwarnung ausgegeben.
Will man alle Warnungen vor möglichen Fehlern unterdrücken, lautet die Kommandozeile:
MASM /W0 *programm*
Nur bei groben Fehlern (MASM-Jargon: severe errors) erfolgt jetzt noch eine Meldung.
Stellt man die Schwelle für Fehlermeldungen mit /W1 ein, produziert der Assembler bereits bei zweideutigen Anweisungen eine Fehlermeldung. Die niedrigste Schwelle wird mit der /W2-Option eingestellt: Anweisungen, die einen ineffizienten Code zur Folge haben, können bereits eine Warnung auslösen.
Wenn man sich sicher ist, daß der Assembler trotz Warnung den richtigen Code produziert, kann man die Warnung ignorieren. Es ist jedoch ein besserer Programmierstil, die Ursache der Warnung zu beseitigen.

Inklusive Codeview

Da es wohl niemand schafft auf Anhieb ein fehlerfreies Programm zu erstellen, gehört zu einem Assembler auch ein guter Debug-

```
                DOSSEG                          ;Segment-Konvention von
                                                ;Microsoft verwenden
                .8086                           ;8088/8086-Befehlssatz
                .MODEL SMALL                    ;Kleines Speichermodell
                .STACK 100H                     ;Stack: 256 Byte
;
cr              equ     0dh                     ;Carriage Return
lf              equ     0ah                     ;Line Feed
ascii_offset_0  equ     '0'                     ;Codierung der Zahl 0 in ASCII
stdaus          equ     1                       ;Handle für Standard-Ausgabe
;
                .DATA                           ;Daten-Segment
string          db      cr,lf,'Uhrzeit: 00:00:00.00',cr,lf  ;Systemzeit
;  Zehnerstelle:        0  0   0000000011111111122  Position der einzelnen
;  Einserstelle:        0  1   2345678901234567890  Byte im String
string_laenge   equ     $-string
;
                .CODE                           ;Daten-Segment
main            label   far
start:          mov     ax, @data               ;ds-Register laden
                mov     ds, ax                  ;über ax
                mov     ah, 2ch                 ;Systemzeit abfragen
                int     21h                     ;ch:h, cl:m, dh:s, dl:h_s
                mov     string+12, ch           ;Speichern der Systemzeit
                mov     string+15, cl           ;im Feld string als
                mov     string+18, dh           ;Dualzahl
                mov     string+21, dl           ;
                mov     di, offset string       ;Startadresse des Feldes in di
                mov     bx, 21                  ;21 Zeichen ohne letztes crlf
schleife1:      mov     al, [bx+di]             ;bx+di --> auf ein Byte des Strings
                call    near ptr dual_ascii     ;Umwandeln Binär -> ASCII
                sub     bx, 3                   ;bx := bx - 3
                cmp     bx, 10                  ;Vergleich mit dem vorletzten
                                                ;Element des Ausgabebereichs
                jnb     schleife1               ;Abbruchbedingung erfüllt?
                call    near ptr ausgabe        ;wenn ja: Erfolgsmeldung
                mov     ax, 4c00h               ;Rückkehr ins DOS
                int     21h                     ;
                retf                            ;Ende des Hauptprogramms
;
ausgabe:        mov     bx, stdaus              ;Handle für Standard-Ausgabe
                mov     cx, string_laenge       ;Länge der Meldung in cx
                mov     dx, offset string       ;Offset der Meldung in dx
                mov     ah, 40h                 ;DOS-Funktion 40h
                int     21h
                retn                            ;Rückkehr
                                                ;
dual_ascii:     push    bx                      ;bx sichern (Zeichen in al)
                xor     ah, ah                  ;ah := 0
                mov     cx, 10                  ;cx := 10
schleife2:      xor     dx, dx                  ;dx := 0
                div     cx                      ;dl := (dx:ax)/10
                add     dl, ascii_offset_0      ;ASCII := Dual + '0'
                mov     [bx+di], dl             ;ASCII-Ziffer im Feld string
                                                ;speichern
                dec     bx                      ;nächste Ziffer
                cmp     ax, 0                   ;Dualzahl konvertiert?
                jnz     schleife2
                pop     bx                      ;bx restaurieren
                retn                            ;Rückkehr
end             start                           ;Einsprungpunkt bei start
```

Bild 3. Vereinfachte Assembler-Direktiven klären den Blick auf das Wesentliche

ger. Vergleicht man einmal die Bedienerfreundlichkeit des mit MS-DOS gelieferten DEBUG.COM und des mit MASM 4.0 gelieferten Symdeb mit dem des Codeview von MASM 5.0, macht man jeweils einen Schritt von der Steinzeit über die Bronzezeit zur Jetztzeit.

Codeview ist – kurz formuliert – ein symbolischer Debugger, der seine Ausgaben in verschiedenen Fenstern vornimmt.

Während man unter DEBUG.COM von Sprungzielen und Variablen nur ihre Adressen bzw. ihre Werte in hexadezimaler (sedezimaler) Notation kennt, spricht man sie bei einem symbolischen Debugger direkt mit ihren jeweiligen Namen an. Will man z. B. den Wert der Variablen string_laenge wissen, gibt man folgenden Befehl ein:

? string_laenge

Auch mit einem symbolischen Debugger will die Fehlersuche gelernt sein: im Handbuch sind fast 250 Seiten Codeview gewidmet.

Codeview kann im Fenster-Modus und im herkömmlichen sequentiellen Modus betrieben werden. Im sequentiellen Modus verhält es sich ähnlich dem älteren Debugger Symdeb. Im Fenstermodus wird der Bildschirm in je ein Quellcode-, ein Register-, ein Dialog- und ein Beobachtungs-Fenster aufgeteilt. Codeview ist komplett menügeführt und kann auch mit der Maus bedient werden. Die mächtigsten Kommandos stehen dem Anwender allerdings im Dialog-Fenster zur Verfügung.

Während des Tests ist mir das Dialog-Fenster unangenehm aufgefallen. Befehle, die man im Dialog-Fenster eingibt, kann man nur äußerst primitiv editieren. Hat man sich bei einem Kommando vertippt, muß man mit der Backspace-Taste alles bis zum Fehler löschen und den Rest des Kommandos frisch eintippen. Hier fühlt man sich noch deutlich an die Steinzeit der Debugging-Technik erinnert.

Positiv zu verbuchen ist, daß man mit Codeview nicht nur Fehler in Assembler-Programmen, sondern auch in compilierten Programmen suchen kann. Um die Fehlersuche im Quellcode vorzunehmen, muß man einiges beachten.

Beim Assemblieren muß MASM 5.0 mit der /ZI-Option gestartet werden, damit er im erzeugten Objektmodul die Zeilennummern und den Quellcode speichert. Frühere MASM-Versionen kennen die /ZI-Option nicht: ältere Programme kann man daher nicht auf Quellcode-Ebene debuggen.

Zum Linken von Programmen, die mit Codeview „entwanzt" werden sollen, benötigt man Microsofts Overlay-Linker ab Version 3.6. Er ist im MASM-5.0-Paket enthalten.

Damit ein symbolischer Debugger richtig arbeiten kann, muß ihm der Quellcode bekannt sein. Gibt man beim Linken die Option /CO an, wird der Quellcode in der EXE-Datei gespeichert. Codeview kann sich aus der so präparierten EXE-Datei alle Informationen holen.

Ist das Programm fehlerfrei, sollte man es ohne die /CO-Option linken, um die Größe der EXE-Datei zu verringern. Vergißt man das, kann sich jeder mit Codeview den Quellcode des Programms anschauen.

Tips zu Codeview

Obwohl der Assembler sowohl das Microsoft Fließkomma-Format als auch das genormte IEEE-Format versteht, erkennt Codeview nur das IEEE-Format.

Makros sollten vor der Fehlersuche expandiert werden, sonst behandelt sie Codeview wie eine normale Anweisung. Fehler in einem Makro blieben unentdeckt.

Codeview eignet sich auch zum Debuggen von Programmen, die mit einem der neueren Microsoft-Compiler erstellt wurden. Auch beim Compilieren muß man die /Zi-Option eingeben.

Nützliche Hilfsprogramme

Neben dem Linker, der Make-Utility und dem Bibliotheks-Manager fügt Microsoft dem Assembler weitere Hilfsprogramme bei. Dazu gehören neben den von der Vorgängerversion her bekannten Utilities EXEPACK und EXEMOD, die Hilfsprogramme SETENV und ERROUT.

EXEPACK komprimiert eine bereits bestehende EXE-Datei. Es optimiert die Tabelle der Relokationseinträge und ersetzt sich wiederholende Datenfolgen durch Wiederholungsanweisungen [1]. Bei den meisten Programmen wird die Größe und die Ladezeit vermindert.
Mit EXEMOD kann man den Header einer EXE-Datei verändern oder anzeigen. Die Größe des Stacks läßt sich mit EXEMOD noch nach dem Linken verändern.
Eine nützliche Utility für Anwender, deren Rechner mit einer Festplatte ausgestattet ist und die mit einer älteren DOS-Version arbeiten, heißt SETENV. Weil das DOS-Environment nur auf 160 Byte voreingestellt ist, kommt man mit dem Set- und dem Path-Kommando schnell an die obere Grenze. Mit dem Hilfsprogramm SETENV vergrößert man das Environment auf z. B. 608 Bytes wie folgt:

```
SETENV COMMAND.COM 608
```

Anwender neuerer DOS-Versionen (ab 3.2) benötigen SETENV nicht. Sie schreiben in die Konfigurationsdatei (CONFIG.SYS) das folgende Kommando:

```
SHELL = COMMAND.COM /E:608 /P
```

Mit dem Hilfsprogramm ERROUT kann man jede gültige DOS-Befehlszeile ausführen und die Standardfehlerausgabe auf eine Datei oder ein Gerät umleiten. Die Standardausgabe wird von ERROUT nicht verändert. Eventuell vorhandene Umleitungsoperatoren für die Standardausgabe bleiben gültig.
Eine Anwendung der Utility könnte z. B. so aussehen:

```
ERROUT /f PRN MASM /ZI PRIM1,,;
```

Das Programm PRIM1 wird assembliert und eventuelle vorhandene Fehlermeldungen über den Drucker ausgegeben. Alle anderen Ausgaben erscheinen auf dem Bildschirm.

Tabelle: Zusammenfassung der MASM-Optionen

Option	Bedeutung
/A	Segmente in alphabetischer Reihenfolge in die Objekt-Datei schreiben
/B*number*	Größe des Puffers für die Quelldatei. Mit *number* verändert man die Anzahl der 1-KByte-Blöcke
/C	Querverweisliste erzeugen (cross-reference)
/D	Schreibt das Ergebnis des ersten Durchgangs in die LST-Datei
/D*symbol*[[= *wert*]]	Wirkt wie eine Equ-Anweisung im Quellcode
/E	Daten und Code erzeugen für 8087-/80287-/80387-Emulatoren
/H	Alle Assembler-Optionen und die Befehlszeilen-Syntax anzeigen
/I*path*	Pfad für Include-Dateien bestimmen
/L	LST-Datei anlegen
/ML	Bei den Namen von Symbolen wird zwischen Groß- und Kleinschreibung unterschieden
/MU	Namen von Symbolen in Großbuchstaben umwandeln
/MX	Bei Public- und External-Namen zwischen Groß- und Kleinschreibung unterscheiden
/N	In der LST-Datei sollen keine Tabellen für Makros, Strukturen, Records, Segmente, Segmentgruppen und Symbole stehen
/P	Überprüft den Code auf Privilegverletzungen. Wichtig für die geschützte Betriebsart der Prozessoren 80286 und 80386
/S	Segmente in der Reihenfolge in die Objekt-Datei schreiben wie sie im Quellcode vorkommen
/T	Unterdrückt die Meldung, daß die Assemblierung erfolgreich verlaufen ist (terse = bündig)
/V	Gibt die Zahl der Symbole und der Zeilen auf dem Bildschirm aus (verbose = geschwätzig)
/W{0|1|2}	• Legt den Fehlermeldungsgrad fest
/X	Bei der bedingten Assemblierung werden auch die Anweisungen in die LST-Datei geschrieben, deren Bedingung nicht erfüllt ist
/Z	Gibt die Zeilen des Quellcodes aus, in denen Fehler entdeckt wurden
/ZD	• Speichert die Zeilennummern in der Objektdatei
/ZI	• Speichert die Symbole und die Zeilennummern in der Objektdatei
Ausgewählte Optionen des Linkers:	
/CO	Die EXE-Datei wird mit Informationen für Codeview versehen
/HE	Liste der Optionen auf dem Bildschirm darstellen
/LI	In der MAP-Datei stehen die Zeilennummern aus dem jeweiligen Quellprogramm, wenn es mit der ZD-Option assembliert wurde
/M	Alle Symbole, die mit der Direktive Public gekennzeichnet sind, werden in der MAP-Datei aufgelistet
/ST:*number*	Größe des Stacks angeben. *number* stellt die Größe des Stacks in Bytes dar.
•:	neue Option
|:	Symbol für oder
{}:	Einer der Werte in der geschweiften Klammer muß eingegeben werden
[[ ]]:	Wert innerhalb der eckigen Klammer ist optional

Für Einsteiger geeignet

Wer im PC-Bereich in Assembler programmieren will, ist mit MASM 5.0 gut bedient. Mit einem Preis, der im „Turbo-Bereich" liegt, dürfte er für viele Anwender erschwinglich sein. Voraussetzung für seine erfolgreiche Anwendung sind allerdings gute Englischkenntnisse.
Trotz der gegenüber seinen Vorgängerversionen erheblich umfangreicheren Dokumentation, reichen die Handbücher zum Erlernen des Programmierens in Assembler nicht aus. Microsoft weist gleich zu Beginn des Programmier-Handbuchs auf mehrere Bücher zum Thema Assembler hin. Darunter sind unseres Wissens folgende in Deutsch erhältlich:

R. Duncan: MS-DOS für Fortgeschrittene [2]

P. Norton: Programmierhandbuch für den IBM PC [3]

Mit MASM 5.0 und der genannten Literatur kann man sich dann tatsächlich zum „Crack" in Assembler entwickeln.

Für Anwender älterer MASM-Versionen dürfte der Umstieg besonders wegen Codeview und den vereinfachten Segment-Direktiven interessant sein.

Literatur

[1] *Smode, Dieter:* Das große MS-DOS-Profi-Arbeitsbuch. Franzis-Verlag, München.
[2] *Duncan, Ray:* MS-DOS für Fortgeschrittene. Vieweg-Verlag, Braunschweig.
[3] *Norton, Peter:* Programmierhandbuch für den IBM PC. Vieweg-Verlag, Braunschweig.

Ulrich Rohde

Ein Alleskönner

Der Star LC-10

Es tut sich was auf dem Druckermarkt. Die preiswerten Drucker werden immer leistungsfähiger. Der Star-Drucker LC-10 ist eines von den neuen Geräten, die gerade auf dem Markt gekommen sind und fast alles können. Er ist für den Privatmann gedacht, der Briefe bester Qualität schreiben will.

Das neue Modell LC-10 von Star mußte bei mir ganz schön etwas leisten. Ich habe es eine Zeit lang als meinen Arbeitsdrucker benutzt, um so auf Schwächen aufmerksam zu werden, die bei einem Kurztest nicht gefunden würden. Ich muß allerdings sagen, daß ich wenig Schwächen gefunden habe. Das Schlimmste vorweg: Das Gerät ist beim Grafikausdruck bei den senkrechten Linien minimal aber sichtbar ungenau. Das liegt an der einfachen Konstruktion, die das Gerät aber auch preiswert macht. Ich kann mit dieser Ungenauigkeit leben.

Das Handbuch

Wenn ein Drucker bei einem Privatmann zuhause steht, dann ist nicht immer gewährleistet, daß professionelle Software vorhanden ist, die so viele Drucker im Einstellmenü besitzt, daß man seinen Drucker ohne Nachdenken anschließen kann. Deshalb muß das Handbuch gut sein, damit man bei der Installation nicht unnötig herumrätseln muß. Das Handbuch für den Star ist offenbar von qualifizierten Leuten geschrieben, die sowohl vom Schreiben, als auch von Druckern und Computern etwas verstehen. Es macht Spaß darin zu lesen. Allerdings ist es keine leichte Kost, es ist ziemlich dicht mit Informationen voll gepackt. *Bild 1* zeigt den Drucker, wie ihn das Handbuch darstellt, wenn es um das Einlegen von Endlospapier geht. Der Drucker ist sehr kompakt gebaut. Die Traktor-Zuführung für Endlos-Papier ist in das Gehäuse integriert.

Papiereinlegen

Mit dem Handbuch war es leicht, das Gerät als erstes für den Gebrauch zu installieren. Es war im Grunde nur das Farbband zu montieren. Das Papiereinlegen ist ebenfalls einfach. Man entfernt die Abdeckteile und legt das Papier in die Stachel-Transporträder ein, wie bei jedem anderen Drucker. Dann allerdings kann man schon die Halbautomatik benutzen, die der LC-10 für seine Freunde bereit hält: Durch Bewegen des Papierfreigabehebels (von der Walze weg)

Bild 1. Schemazeichnung des Druckers aus dem Handbuch

kann man bei eingeschaltetem Drucker bewirken, daß das eben eingelegte Endlospapier automatisch in die richtige Druckposition gebracht wird. Drehen am Walzenknopf erübrigt sich. Eigentlich ist diese Automatik für den gemischten Betrieb „Einzelblatt und Endlos" gedacht. Der ist wirklich bequem durchzuführen. Drückt man zwei bestimmte Tasten auf dem Bedienertastenfeld gleichzeitig, dann wird das Endlospapier automatisch aus dem Walzenbereich nach rückwärts herausgefahren. Allerdings nur, wenn es nicht mehr als etwa eine halbe Seitenlänge vorwärts herausschaut, sagt das Handbuch. Ich habe es ausprobiert, tatsächlich wird es, wenn schon mehr herausschaut, nicht mehr ganz zurückgezogen. Ich habe aber einfach nochmals den Parktransportvorgang ausgelöst, und da fuhr dann das Papier im zweiten Schritt das letzte Stück auch noch zurück. In dieser Stellung kann man den Wahlhebel für die Papierart auf Einzelblatt stellen, die Papierführung senkrecht und schon ist man im Einzelblattbetrieb. Ich muß sagen, sehr bequem. Diese Betriebsart funktioniert ebenfalls prächtig. Man muß das Einzelblatt nur gut in den Schacht herabführen, damit die Walze greifen kann. Um nun das Papier richtig zu justieren, gibt es die Möglichkeit, den Transport in Mikroschritten vozunehmen. Und zwar vorwärts und rückwärts. Die einzige Schwierigkeit ist dabei, daß man sich die vielen Tastenkombinationen merken muß, die alle diese Einstellungen bewirken. Aber das ist vielleicht nur für einen Tester eine Schwierigkeit, der die unterschiedlichsten Modelle inzwischen bedient hat und immer wieder neue Bedienvorschriften findet. Allerdings hat Star etwas Gutes für vergeßliche Menschen getan: Dem Gerät liegt eine Karte bei, die eine Kurzbeschreibung der Funktionen enthält. *Bild 2* zeigt die Karte.

Im praktischen Betrieb

An die Eigenheiten der Bedienung eines Gerätes gewöhnt man sich schnell. Schlimmer ist es, wenn eine Eigenschaft im Betrieb fehlen würde, die man unbedingt benötigt. Da kann beim LC-10 aber kaum etwas fehlen. Die Einstellmöglichkeiten per Konsole und per Software sind reich. Es sind 14 Zeichensätze wählbar. Drei Schriftarten gibt es, die auch kursiv gedruckt werden können. Der Drucker besitzt zwei Druckarten. Im Draft-Betrieb ist er schnell und nadelt jedes Zeichen nur einmal auf das Papier. Im NLQ-Modus schreibt er jedes Zeichen zweimal. Der Star fährt dazu jeweils die ganze Zeile zweimal von der gleichen Seite an. Das erleichtert die Justage der Nadelpunkte gegeneinander. *Bild 3* zeigt die Schriften die in NLQ und Draft zur Verfügung stehen. Es gibt eine Fülle von Befehlen, die der Drucker aus dem Bytestrom entnimmt, den er drucken soll. Es sind dies die Escape-Sequenzen, die bei jedem Drucker Steuerfunktionen auslösen. Was die Sache etwas unübersichtlich macht: Der Drucker besitzt zwei Hauptbetriebsmodi, einmal als Standard den Epson-LX-800-Modus und einmal den IBM-Proprinter-Modus. Viele Befehle besitzen nun

LC-10 Kurzbeschreibung star

Die Bedienerkonsole

Durch einen Knopfdruck auf der Bedienerkonsole werden eine Vielzahl von Funktionen des Druckers aktiviert. Wir beschreiben die Tasten und Anzeigen auf der Bedienerkonsole des Druckers von rechts nach links. Insgesamt stehen hier zehn Anzeigen und vier Tasten zur Verfügung.

POWER-ANZEIGE – leuchtet bei eingeschaltetem Drucker orange, oder blinkt, wenn das Papierende erreicht ist.

ON LINE-TASTE und ANZEIGE – leuchtet gelb, sobald der Drucker für den Datenempfang bereit ist. Ein Drücken der ON LINE-Taste schaltet den Drucker ON LINE bzw. OFF LINE.

PAPER FEED-TASTE – schiebt das Papier jeweils um eine Zeile vor, sofern die ON LINE-Anzeige nicht leuchtet.

PRINT PITCH-TASTE – wählt einen der verfügbaren Zeichenabstände.

PITCH-ANZEIGEN – zeigt den eingestellten Zeichenabstand an (siehe Handbuch).

NLQ TYPE STYLE TASTE – schaltet die Schriftarten im OFF LINE Betrieb um. Eine Draft, drei Roman und drei Kursiv-Schriftarten sind wählbar.
Orator kann in zwei Versionen gewählt werden (DIP-Schalter 1-3).

Viele Funktionen sind nicht direkt auf der Bedienerkonsole beschrieben. Einige werden durch Drücken einer bestimmten Taste beim Einschalten des Druckers aktiviert, andere durch gleichzeitiges Drücken von zwei Tasten.

IM OFF-LINE-BETRIEB NACH DEM EINSCHALTEN

Folgende Tastaturfunktionen sind nur im OFF LINE-Betrieb des Druckers verfügbar.
Leuchtet die ON LINE-Anzeige, drücken Sie bitte die ON LINE-Taste um den Drucker OFF LINE zu schalten.

Blattvorschub – wirft bei Einzelblattbetrieb das Blatt aus; schiebt bei Endlospapier das Blatt zum nächsten Blattanfang vor. Halten Sie die PAPER FEED Taste und drücken Sie die ON LINE Taste.

MIKRO-PAPIERTRANSPORT
Hiermit wird ein bidirektionaler Mikro-Papiertransport durchgeführt (mit einer automatischen Blattzuführung ist diese Betriebsart nicht verfügbar).

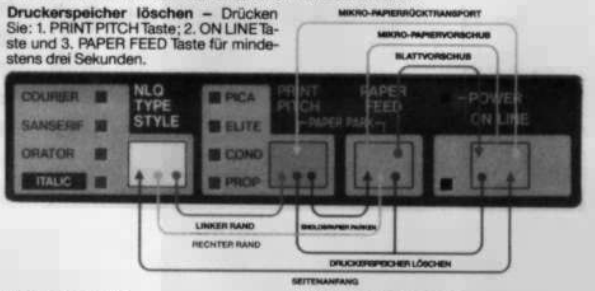
MIKRO-PAPIERVORSCHUB
Halten Sie die ON LINE-Taste und drükken Sie die PAPER FEED-Taste.

MIKRO-PAPIERRÜCKTRANSPORT
Halten Sie die ON-LINE-Taste und drükken Sie die PAPER FEED-Taste.

RANDEINSTELLUNG
Eine wichtige Funktion in der Textverarbeitung.

LINKER RAND
Zum Setzen des linken Randes halten Sie die NLQ TYPE STYLE-Taste und drücken die PRINT PITCH-Taste . . . dabei bewegt sich der Druckkopf im PICA-Abstand. Lassen Sie die PRINT PITCH-Taste los, sobald sich der Druckkopf an der gewünschten linken Randposition befindet.

Druckerspeicher löschen – Drücken Sie: 1. PRINT PITCH Taste; 2. ON LINE Taste und 3. PAPER FEED Taste für mindestens drei Sekunden.



RECHTER RAND
Zum Setzen des rechten Randes halten Sie die NLQ TYPE STYLE-Taste und drükken die PAPER FEED-Taste . . . dabei bewegt sich der Druckkopf im PICA-Abstand. Lassen Sie die PAPER FEED-Taste los, sobald Sie die gewünschte rechte Randposition erreicht haben.

SEITENANFANG (TOF)
Der Seitenanfang wird wie folgt eingestellt: Drücken Sie die PAPER FEED-Taste bis zur gewünschten Position. Drücken Sie dann die NLQ TYPE STYLE und ON LINE-Tasten zum Speichern der neuen Position.

ENDLOSPAPIER PARKEN
– Siehe andere Blattseite.

● Taste halten ▼ Taste drücken

BEIM EINSCHALTEN DES DRUCKERS

Beim Einschalten der Stromzufuhr des Druckers kann auf folgende Betriebsarten zugegriffen werden.

SELBSTTEST
Die Ausgabe eines kompletten Selbsttestausdruckes erfolgt durch Halten der Taste PAPER FEED oder ON LINE. Die Betriebsbereitschaft des Druckers wird dadurch bestätigt.
Selbsttest 1 – Lange Version
Selbsttest 2 – Kurze Version

HEX DUMP
Halten Sie zur Einleitung des Hex Dump-Modus gleichzeitig die Tasten PAPER FEED und PRINT PITCH. Sie erhalten auf diese Weise einen exakten Ausdruck der vom Drucker benutzten hexadezimalen Codes; Problemlösungen werden so vereinfacht. Dieser Modus wird durch Ausschalten des Druckers beendet.

PANEL-BETRIEB
Der Drucker verfügt über mehrere Software-Steuermöglichkeiten. Wenn Sie jedoch in einer Betriebsart drucken und dabei die Steuercodes umgehen möchten, wählen Sie den PANEL-BETRIEB. Der PANEL-BETRIEB wird durch Halten der NLQ TYPE STYLE-Taste und/oder der PRINT PITCH-Taste aktiviert. Diese Betriebsart bleibt bis zum Ausschalten des Druckers aktiv.

Weitere Informationen finden Sie in Ihrem Drucker Handbuch.

Änderungen vorbehalten.

star der ComputerDrucker

Bild 2. Eine gute Idee: Kurzbeschreibung der Bedienung des Druckers

Varianten, die für den einen oder anderen der beiden Modi gültig sind. Es ist deshalb hoffnungslos, alles aufzuzählen. Programme mit Epson-Interface werden laufen und den Drucker richtig steuern – vorausgesetzt, er ist in der richtigen Einstellung gewesen. Nach dem Einschalten befindet sich der Drucker zunächst in der Einstellung, die durch DIP-Schalter vorgegeben ist. Davon kann man ihn durch Escape-Sequenzen abbringen. Allerdings kann man auch beim Einschalten bestimmen, daß man den Drucker von seiner Konsole her einstellen möchte. Dann ignoriert er etwa vom Computer zugesandte Steuercodes. So kann man Textprogramme überlisten, wenn einem Schriftart und andere Einstellungen nicht passen.

Der Drucker wurde an mehreren PC-Kompatiblen betrieben. Dazu habe ich ihn an den Schaltern für den Proprinter-Betrieb präpariert. IBM hat dafür zwei Zeichensätze entworfen, von welchen der zweite oberhalb von 127 nützliche Sonderzeichen besitzt (Umlaute und andere Sonderzeichen). Wenn man da die richtige Software besitzt, kann man einerseits gut Programme in den verschiedensten Programiersprachen ausdrucken, weil sie meist nur Zeichen unter 128 benutzen. Andererseits kann man – und dazu benötigt man die richtige Software – deutsche Texte ohne Umschalten des Druckers bearbeiten. Es müssen nur die Umlaute als passende Codes ausgegeben werden. Die einzige Schwierigkeit hier war bei mir die richtige Einstellung zu finden. Meine Software war einstellbar, mein Drucken auch, und das ergab überwältigend viele Irrtumsmöglichkeiten.

```
10 'BEISPIELPROGRAMM SCHRIFTARTEN
20 LPRINT "Die Roman Schriftarten des Druckers":GOSUB 50
30 LPRINT "Und nun in Kursiv"
40 LPRINT CHR$(28);"4";:GOSUB 50:END
50 LPRINT CHR$(27);"I";CHR$(0);"Draft"
60 LPRINT CHR$(27);"x";CHR$(1);"NLQ"
70 LPRINT CHR$(27);"k";CHR$(1);"NLQ Sanserif"
80 LPRINT CHR$(27);"k";CHR$(0);"NLQ Courier"
90 LPRINT CHR$(27);"k";CHR$(2);"NLQ Orator 1"
100 LPRINT CHR$(27);"k";CHR$(3);"NLQ Orator 2"
110 LPRINT CHR$(27);"@";
120 RETURN
```

```
Die Roman Schriftarten des Druckers
Draft
NLQ
NLQ Sanserif
NLQ Courier
NLQ ORATOR 1
NLQ Orator 2
Und nun in Kursiv
Draft
NLQ
NLQ Sanserif
NLQ Courier
NLQ ORATOR 1
NLQ Orator 2
```

Bild 3. Schriftproben

Drucker:	Serieller Punktmatrixdrucker	
Druckkopf:	9 Nadeln	
Druckgeschwindigkeit:	144 Zeichen pro Sekunde (im Draft-Elite-Betrieb)	
	120 Zeichen pro Sekunde (im Draft-Pica-Betrieb)	
	30 Zeichen pro Sekunde (im NLQ-Pica-Betrieb)	
Druckerspeicher:	4 KByte, 1 Zeile bei Download Betrieb	
Papiervorschub-geschwindigkeit:	2.7 inch pro Sekunde (beim Blattvorschub)	
Papiervorschub:	Traktor- oder Friktionsbetrieb	
Druckrichtung:	Bidirektional mit Druckwegoptimierung	
	Unidirektional im Grafik-Betrieb und NLQ-Betrieb	
Zeichensätze:		
Standard Zeichensatz	96 ASCII Zeichen	
IBM Zeichensatz	244 Zeichen (ASCII Sonderzeichen, Blockgrafik, Internationale Zeichen)	
Internationale Zeichensätze:	14 Sätze	
Schriftarten:		
Draft:	Draft; Draft kursiv (nur über Steuerbefehl)	
NLQ:	Courier, Sanserif, Orator und deren Kursivausdruck	
Sonstiges:	Doppelt/vierfach hohe und breite Zeichen	
Sonstige Zeichen:	192 freidefinierbare Draft-Zeichen oder	
	78 freidefinierbare NLQ-Zeichen	
Zeichenmatrix:	9 × 9 Punkte bei Draft Pica	
	18 × 23 Punkte bei Courier und Orator Pica	
	18 × 18 Punkte bei Sanserif Pica, Elite	
	12 × 11 Punkte bei IBM Blockgrafik, Pica	
	18 × 19 Punkte bei Courier und Orator Elite	
	18 × 12 Punkte bei Pica Schmalschrift	
	18 × 10 Punkte bei Elite Schmalschrift	
Grafikbetrieb:	8 × 60 Punkte/inch Normaldichte	
	9 × 60 Punkte/inch Normaldichte	
	8 × 120 Punkte/inch Doppeldichte	
	9 × 120 Punkte/inch Doppeldichte	
	8 × 120 Punkte/inch Doppeldichte, doppelte Geschwindigkeit	
	8 × 240 Punkte/inch Vierfachdichte	
	8 × 80 Punkte/inch Bildschirmgrafik Typ I	
	8 × 90 Punkte/inch Bildschirmgrafik Typ II	
	8 × 72 Punkte/inch Plottergrafik	
Zeilenabstand:	1/6 inch Standard	
	1/8 inch bzw. *n*/72 oder *n*/216 inch programmierbar	
Spaltenbreite:	Pica Breitdruck	40 Zeichen/Zeile
	Elite Breitdruck	48 Zeichen/Zeile
	Pica Schmalschrift-Breitdruck	68 Zeichen/Zeile
	Elite Schmalschrift-Breitdruck	80 Zeichen/Zeile
	Pica	80 Zeichen/Zeile
	Elite	96 Zeichen/Zeile
	Pica Schmalschrift	137 Zeichen/Zeile
	Elite Schmalschrift (nur Standard)	160 Zeichen/Zeile
	Proportionaler Ausdruck	variabel
Sonderfunktionen:	Schönschriftausdruck (NLQ)	
	Halbautomatischer Einzug eines Blattes	
	Papierabrißkante	
	Hex Dump und 2 Selbsttestfunktionen	
	Perforationssprung; Papierrücktransport	
	Endlospapier "parken"; Panel Betriebsarten	
	Leicht zugängliche DIP-Schalter	
	Benutzerdefinierbare Zeichen in NLQ und Draft Ausdruck	
	Makroanweisungen; Grafik mit ultrahoher Auflösung	
	Vertikal- und Horizontaltabulatoren	
	Automatischer Einzelblatteinzug (Option)	
Druckerpapier:	Einzelblätter 14-21 cm breit, Endlospapier 10-25 cm breit	
Papierstärke:	einfach 0.07-0.10 mm, max. dreifach max. 0.28 mm	
Kopien:	1 Original und 2 Kopien	
Druckerabmessungen:	Höhe 108 mm, Breite 384 mm, Tiefe 287 mm	
Gewicht:	4,7 kg	
Spannungsversorgung:	220 Volt ± 10%; 50/60 Hz; max. 60 W	
Umgebungstemperatur:	5 bis 40 °C	
Luftfeuchtigkeit:	10-80 % (nicht kondensierend)	
Farbband:	Original Star Farbbandkassette	
Standzeit Farbband:	1 Mio. Draft-Zeichen	
Standzeit Druckkopf:	200 Mio. Punkte	
Parallel Interface:	Centronics kompatibel, 7 oder 8 Bit Interface	

Bild 4. Die technischen Daten

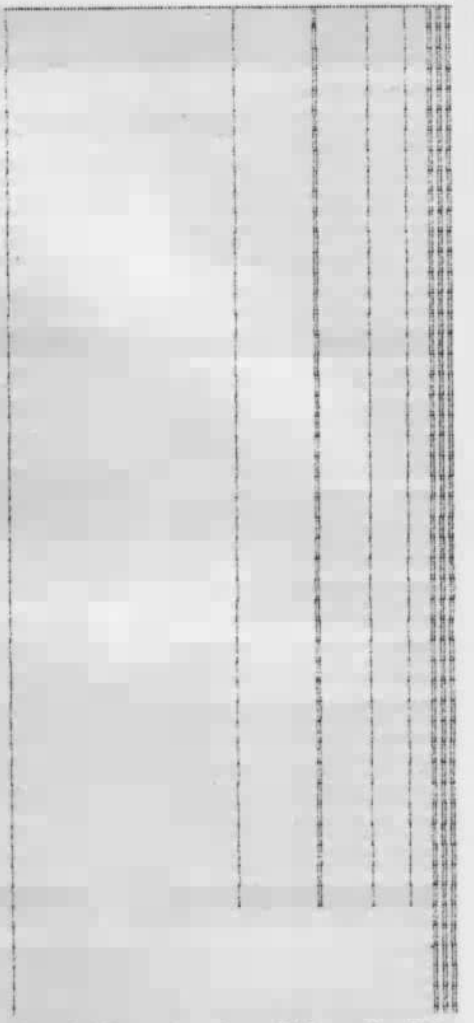


Bild 5. Die senkrechten Linien mit „Toleranzen"

Anpassungs-Schwierigkeiten

Es kann unterstellt werden, daß die korrekte Anpassung des Druckers der schwierigste Teil beim Betrieb ist. Bei so viel Einstellmöglichkeiten, die einander sogar überlagern können, wird es für Gelegenheitseinsteller viele Fehlermöglichkeiten geben. Wer die aber überwunden hat, kann mit diesem Drucker sehr viel machen. Die technischen Daten (*Bild 4*) sprechen für sich. Die Bedienung ist bequem, man kann die Halbautomatik bestens nutzen. Zum Beispiel hatte ich mich beinahe damit abgefunden, daß die Abreißkante für das Papier nicht mit der Seitenlage übereinstimmte, die ich für richtig hielt. So wäre immer eine Seite beim Beenden eines Schriftstückes auf Endlospapier verloren gewesen. Aber wenn man das Papier mit dem Zeilenvorschub abreißgerecht vorfährt, kann man nach dem Abreißen mit der Parkautomatik das Papier wieder zurückfahren und anschließend in die Startposition bringen. Sehr bequem für einen Drucker dieser Preisklasse (etwa 550.– DM).

Wilfried Voss

Bildschirmfunktionen für Turbo-C

Die meisten Programme bieten heute dem Anwender eine Benutzerführung im Dialog – Software-Anbieter müssen dieser Entwicklung Rechnung tragen. Dem Turbo-C-Programmierer werden hier deshalb Funktionen zur Bildschirm-Ein- und -Ausgabe sowie zur Cursorsteuerung vorgestellt.

Das Programmpaket besteht aus verschiedenen Dateien, die nachfolgend aufgeführt sind. Sie werden einfach zum eigenen Anwenderprogramm gebunden, in das sinnvollerweise auch die Header-Dateien keyboard.h (*Bild 1*) und screen.h (*Bild 2*) per #include aufgenommen werden.

Die Datei „screen.c" *(Bild 3)* enthält die allgemeinen Funktionen für die Cursormanipulation und Bildschirmsteuerung einschließlich einer schnellen Bildschirmausgabe. Mit „editline.c" steht ein komfortabler Zeileneditor zur Verfügung, der in eigene Programme eingebunden werden kann. In beide Quelldateien werden außer den Standard-Header-Dateien des Compilers zwei weitere zur Deklaration von Funktionen und Konstanten aufgenommen: „keyboard.h" enthält eine Auflistung verschiedener Tastaturcodes (Cursor-, Steuer- und Funktionstasten); in „screen.h" sind Bildschirm-Adressen und -Formate sowie Farbattribute definiert.

Die Cursorfunktionen

Die Funktionen zur Cursormanipulation verwenden in erster Linie sogenannte Escape-Sequenzen, wie sie in der Technical Reference des IBM-PC beschrieben sind. Diese Sequenzen werden einfach als Zeichenkette mit dem printf-Befehl ausgegeben. Wertangaben für Zeilen und Spalten beginnen immer bei Null. Zum Beispiel wird die zehnte Zeile mit zeile=9; festgelegt.

```
void cursor_position(zeile, spalte);
  int zeile, spalte;
```

Der Cursor wird auf die angegebene Zeile und Spalte positioniert. Der nächste auszugebende Text beginnt an dieser Stelle.

```
void save_cursor();
```

Die aktuelle Cursorposition (Zeile, Spalte) wird zwischengespeichert.

```
void restore_cursor();
```

Die zuletzt zwischengespeicherte Cursorposition wird restauriert.

Allgemeine Bildschirmfunktionen

Zunächst zwei Funktionen zum Löschen des Bildschirms bzw. einer Bildschirmzeile.

```
void cls();
```

Der Bildschirm wird gelöscht, wiederum durch Verwendung der erwähnten Escape-Sequenzen. Bildschirm-Attribute werden nicht gesetzt.

```
void del(zeile);
  int zeile;
```

Die angegebene Zeile wird gelöscht. Bildschirm-Attribute werden auch hier nicht gesetzt.

Die nachfolgenden Funktionen sind reine C-Programme, mit denen es auch möglich ist, Farben und andere Attribute – etwa Blinken und High Intensity – zu setzen. In der Datei „screen.h" sind die Bildschirm-Attribute definiert.

Schnelle Zeichenausgabe

Trotz der kurzen Ausführungszeit von Turbo-C-Programmen gestaltet sich die Ausgabe von Zeichen auf dem Bildschirm mit dem printf-Befehl immer noch relativ langsam, insbesondere bei Ausgabe eines kompletten Bildschirmformates. Die Ursache für diese Einschränkung ergibt sich durch die Standardausgabe über den DOS-Interrupt 010h. Als Alternative dazu bietet es sich an, die Zeichen direkt in den Bildwiederholspeicher zu schreiben, was einen erheblichen Geschwindigkeitsgewinn mit sich bringt.

Besonders zu beachten ist dabei die Definition der Konstanten SCHNEE in der Datei screen.h: Beim direkten Schreiben in den Bildspeicher muß bei Verwendung einer Monochrom-Videokarte sowie dem Farbgrafik- oder Hercules-Videoadapter die Vertikal-Austastlücke des Bildsignals abgefragt

```
/* Datei : keyboard.h */
/* Keyboard - Codes   */

#define BACKSPACE       8
#define TABULATOR       9
#define CR             13
#define ESC            27
#define BLANK          ' '

/* Cursorblock : Tastatur liefert 00 + Code */

#define FUNKTION       00

#define LEFT           75
#define RIGHT          77
#define UP             72
#define DOWN           80
#define HOME           71
#define END            79
#define PAGE_UP        73
#define PAGE_DOWN      81
#define INSERT         82
#define DELETE         83

/* Definition der Funktionstasten */

#define F1             59
#define F2             60
#define F3             61
#define F4             62
#define F5             63
#define F6             64
#define F7             65
#define F8             66
#define F9             67
#define F10            68

#define SHIFT          25
/* Anwendung z.B.: SHIFT+F2 */
#define ALT            45
/* Anwendung z.B.: ALT+F10  */
```

Bild 1. In der Datei „keybord.h" werden die Tastencodes für Cursor-, Steuer- und Funktionstasten definiert

werden. Damit wird verhindert, daß bei Bildschirmausgaben Störungen auf dem Schirm („Schnee") zu beobachten sind. Bei Verwendung einer EGA-Karte ist dies z. B. nicht notwendig. Ist in der Datei screen.h die Zeile „#define SCHNEE" enthalten, wird in den entsprechenden Programmteilen mit der #ifdef-Anweisung die Abfrage der Austastlücke eingesetzt. Ist die Abfrage nicht nötig, so genügt es, diese Zeile zu löschen oder durch Markierung als Kommentar ungültig zu machen.

```
void delete(zeile, spalte, attribut, anzahl);
  int zeile, spalte;
  unsigned char attribut;
  int anzahl;
```

Ein Ausschnitt einer Bildzeile wird gelöscht, d. h. ab der gewählten Position (Zeile und Spalte) wird eine bestimmte Anzahl Zeichen gelöscht und mit dem Bildschirmattribut versehen. Ist diese Anzahl größer als der Rest der Zeile, dann wird am Anfang der folgenden Zeile weiter gelöscht. Diese Funktion arbeitet unabhängig von der aktuellen Cursorposition auf dem Bildschirm.

```
/* Datei : screen.h                          */
/* Header-File zu Bildschirmfunktionen       */

#define BILDSCHIRM      0xB8000000  /* Fuer Monochrom auf 0xB0000000 setzen*/
#define BILDSCHIRMPORT  0x3DA       /* Fuer Monochrom auf 0x3BA      setzen*/
#define SCHNEE
#define ZEILEN          25
#define SPALTEN         80
#define TAB              8      /* Anzahl Zeichen fuer Tabulator    */

/* Bildschirmattribute                       */

#define BLAU    1       /* Farben            */
#define GRUEN   2
#define ROT     4
#define WEISS    ROT+GRUEN+BLAU
#define HINTERGRUND     16      /* Anwendung z.B.: HINTERGRUND*BLAU */
#define BLINKEN         128
#define INTENSITY       8

/* Funktionsdeklarationen                    */

void    cls();                          /* Bildschirm loeschen          */
void    cursor_position( int, int );    /* Cursor positionieren         */
void    del( int );                     /* Zeile loeschen               */
void    save_cursor();                  /* Cursor zwischenspeichern     */
void    restore_cursor();               /* Cursor zurueckholen          */
void     write_line( char *, unsigned char, int, int );
void     clear_screen( unsigned char );
void     delete( int, int, unsigned char, int );

/****************** Ende ******************/
```

Bild 2. Wie „keybord.h" sollte auch diese Datei („screen.h") durch #include in das Hauptprogramm aufgenommen werden

```
/* Dateiname : screen.c                      */
/* Funktion  : Allgemeine Bildschirmfunktionen */

#include <stdio.h>
#include <string.h>
#include <dos.h>
#include "screen.h"
#include "keyboard.h"

/* Bildschirm loeschen ohne Attribute        */
void cls()
{ printf( "\033[2J" ); }

/* Cursor positionieren                      */
void cursor_position( zeile, spalte )
int zeile, spalte;
{
 if ( zeile <= 0 ) zeile = 0;
 zeile++;
 if ( spalte <= 0 ) spalte = 0;
 spalte++;
 printf( "\033[%d;%dH", zeile, spalte );
}

/* Zeile loeschen ohne Attribute             */
void del( zeile )
int zeile;
{ printf( "\033[%d;1H\033[K", zeile ); }

/* Cursor zwischenspeichern                  */
void save_cursor()
{ printf( "\033[s" ); }

/* Cursor wieder zurueckholen                */
void restore_cursor()
{ printf( "\033[u" ); }


/* 1 Bildschirmzeile loeschen mit Attributen */
void delete( zeile, spalte, attribut, anzahl )
int  zeile, spalte;
unsigned char attribut;
int  anzahl;
{
 char far *bildschirmadresse;
 int  counter;

 (long)bildschirmadresse = BILDSCHIRM + zeile*80*2 +
 spalte*2;
 disable();
 for( counter=0; counter<anzahl; counter++ )
 {
#ifdef SCHNEE
        while(( inportb( BILDSCHIRMPORT ) & 1 ));
        while( !( inportb( BILDSCHIRMPORT ) & 1 ));
#endif
        *bildschirmadresse++ = BLANK;
        *bildschirmadresse++ = attribut;
 }
 enable();
} /* end delete */


/* Bildschirm loeschen mit Attributen        */

void clear_screen( attribut )

unsigned char attribut;

{
 int  zeile;

 for ( zeile=0; zeile<ZEILEN; zeile++ )
 {
        delete( zeile, 0, attribut, SPALTEN );
 }
} /* end clear_screen */

/* Eine Zeile in Bildschirmwiederholspeicher schreiben */

void write_line( zeileninhalt, attribut, zeile, spalte )

char *zeileninhalt;
unsigned char attribut;
int  zeile, spalte;
{
char far *bildschirmadresse;
int  zaehler, anfang, i;

 zaehler = 1;
 /* Aktuelle Bildschirmadresse berechnen */
 (long)bildschirmadresse = BILDSCHIRM + zeile*80*2 +
 spalte*2;
 disable();
 /* Zeileninhalt ausgeben */
 while ( *zeileninhalt != 0 )
 {
        if ( (zaehler+spalte) >= SPALTEN+1 ) break;
        /* Tabulator als Anzahl Blanks darstellen */
        if ( *zeileninhalt == TABULATOR )
        {
                if ( zaehler == 0 ) anfang = 0;
                else anfang = zaehler % TAB;

                for ( i=anfang; i<TAB+1; i++ )
                {
#ifdef SCHNEE
                    while(( inportb( BILDSCHIRMPORT ) & 1 ));
                    while( !( inportb( BILDSCHIRMPORT ) & 1 ));
#endif
                    *bildschirmadresse++ = 0x20;
                    *bildschirmadresse++ = attribut;
                    zaehler++;
                }
                zeileninhalt++;
        }
        else
        {
#ifdef SCHNEE
                while(( inportb( BILDSCHIRMPORT ) & 1 ));
                while( !( inportb( BILDSCHIRMPORT ) & 1 ));
#endif
                *bildschirmadresse++ = *zeileninhalt++;
                *bildschirmadresse++ = attribut;
                zaehler++;
        }
 }
 enable();
}/* end write_line */
```

Bild 3. Bildschirm oder Zeile löschen, Cursor positionieren oder Zeichenketten ausgeben: Mit diesen Funktionen ist das alles kein Problem

```
/* Datei     : editline.c              */
/* Funktion  : Eine Zeile editieren    */

#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <mem.h>
#include "screen.h"
#include "keyboard.h"
int editline( string, attribut, zeile, spalte, max_laenge )
char *string;
unsigned char attribut;
int  zeile, spalte, max_laenge;
{
       char *stringpointer, *source, *destination;
       int  character, returncode, insert, erste_spalte;

 if( max_laenge == 0 ) max_laenge = SPALTEN - spalte;
       else max_laenge--;
 delete( zeile, spalte, attribut, max_laenge+1 );
 insert = 0;
 erste_spalte = spalte;
       stringpointer  = string + strlen( string );
       write_line( string, attribut, zeile, spalte );
       spalte = spalte + strlen( string );
 cursor_position( zeile, spalte );
     while (1)
     {
      returncode = 0;
            character = getch();
      if ( character == CR )
      {
             returncode = CR;
             break;
      }
      switch ( character )
      {
      case BACKSPACE:
             if ( stringpointer > string )
             {
                    source = stringpointer--;
                    destination = stringpointer;
                    memcpy( destination, source,
                                   strlen( stringpointer ) );
                    spalte--;
                    cursor_position( zeile, spalte );
                    delete( zeile, erste_spalte+strlen(string),
                                                  attribut,1 );
                    write_line( string, attribut, zeile,
                                          erste_spalte );
             }
             break;
    case ESC:
            stringpointer = string;
            strnset( stringpointer, BLANK, strlen( stringpointer ));
            spalte = erste_spalte;
            cursor_position( zeile, spalte );
            write_line( string, attribut, zeile, erste_spalte );
            *stringpointer = 0;
            break;
    case FUNKTION:
            character = getch();
            switch ( character )
            {
            case LEFT:
                   if ( stringpointer > string )
                   {
                          stringpointer--;
                          spalte--;
                          cursor_position( zeile, spalte );
                   }
                   break;
            case RIGHT:
                   if ( *stringpointer != 0 )
                   {
                          stringpointer++;
                          spalte++;
                          cursor_position( zeile, spalte );
                   }
                   break;
            case INSERT:
                   if ( insert == 0 ) insert = 1;
                   else insert = 0;
                   break;
            case DELETE:
                   if ( *stringpointer != 0 )
                   {
                          source = stringpointer+1;
                          destination = stringpointer;
                          memcpy( destination, source,
                                        strlen(stringpointer));
                          delete( zeile, erste_spalte
                                 +strlen(string), attribut, 1 );
                          write_line( string, attribut, zeile,
                                                erste_spalte );

                   }
                   break;
            case HOME:
                   stringpointer = string;
                   spalte = erste_spalte;
                   cursor_position( zeile, spalte );
                   break;
            case END:
                   while ( *stringpointer != 0 )
                   {
                          stringpointer++;
                          spalte++;
                   }
                   cursor_position( zeile, spalte );
                   break;
            default:
                   returncode = character;
            } /* end switch */
            break;
            default:
            if ( character >= BLANK )
            {
                   if ( (insert == 1 || *stringpointer == 0)
                   && strlen(string) <=  max_laenge )
                   {
                      source = stringpointer;
                      destination = stringpointer+1;
                      memcpy( destination, source,
                                   strlen( stringpointer )+1 );
                      *stringpointer = (char)character;
                      write_line( string, attribut,
                                        zeile, erste_spalte );

                      stringpointer++;
                      spalte++;
                      cursor_position( zeile, spalte );
                   }
                   if ( insert == 0 && strlen(stringpointer) > 0 )
                   {
                          *stringpointer++ = (char)character;
                      write_line( string, attribut,
                                        zeile, erste_spalte );

                      spalte++;
                      cursor_position( zeile, spalte );
                   }
             }
        } /* end switch */
        if ( returncode ) break;
       } /* end while */
 return( returncode );
} /* end editline */
```

Bild 4. Zeichenketten lassen sich mit diesem Editor komfortabel ändern oder neu eingeben

```
/* Demo-Programm zu Bildschirmfunktionen */

#include <stdio.h>
#include <string.h>
#include "screen.h"
#include "keyboard.h"

main()
{
 char string[81];
 unsigned char attribut;
 int  zeile, spalte, returncode;

 strcpy( string, "Das ist ein Eingabestring" );

 attribut = HINTERGRUND*BLAU + WEISS;
 clear_screen( attribut );

 spalte = 35;
 for( zeile=5; zeile<20; zeile++ )
        write_line( "Demo", attribut, zeile,
        spalte );

 delete( 10, 0, attribut, 80 );
 attribut = HINTERGRUND*ROT + WEISS;

 while( 1 )
 {
        zeile = 22; spalte = 25;
        returncode = editline( string, attribut,
        zeile, spalte, 30 );

        cursor_position( 3, 30 );
        switch( returncode )
        {
        case F1:
                printf( "Funktionstaste 1 " );
                break;
        case F2:
                printf( "Funktionstaste 2 " );
                break;
        case F3:
                printf( "Funktionstaste 3 " );
                break;
        case F4:
                printf( "Funktionstaste 4 " );
                break;
        case F5:
                printf( "Funktionstaste 5 " );
                break;
        case F6:
                printf( "Funktionstaste 6 " );
                break;
        case F7:
                printf( "Funktionstaste 7 " );
                break;
        case F8:
                printf( "Funktionstaste 8 " );
                break;

        case F9:
                printf( "Funktionstaste 9 " );
                break;
        case F10:
                printf( "Funktionstaste 10" );
                break;
        case PAGE_UP:
                printf( "Page up          " );
                break;
        case PAGE_DOWN:
                printf( "Page down        " );
        }/* end switch */
 if( returncode == CR ) break;
 }/* end while */

}/* end main */
```

Bild 5. Als Starthilfe: Ein kleines Demonstrationsprogramm

```
void clear_screen(attribut);
unsigned char attribut;
```

Der komplette Bildinhalt wird gelöscht und mit dem gewählten Attribut versehen. Realisiert ist dieser Programmteil durch 25faches Aufrufen der delete-Funktion.

```
void write_line(zeileninhalt, attribut,
                                zeile, spalte);
 char *zeileninhalt;
 unsigned char attribut;
 int zeile, spalte;
```

Die Zeichenkette „zeileninhalt" wird ab der gewählten Position (Zeile und Spalte) mit dem Bildschirm-Attribut ausgegeben. Berücksichtigt wird dabei auch das Tabulatorzeichen (\0x09), das durch Ausgabe einer berechneten Anzahl von Leerzeichen dargestellt wird. Die Ausgabe erfolgt nur in der angegebenen Zeile, d. h. bei Überschreitung des Zeilenendes wird die Funktion abgebrochen. Auch diese Funktion ist unabhängig von der aktuellen Cursorposition.

Der Zeileneditor

Dem geplagten DOS-Anwender sei versichert, daß dieser Editor *(Bild 4)* nichts mit dem edlin-Kommando zu tun hat: Die Funktion „editline.c" arbeitet völlig bildschirmorientiert und bietet verschiedene Möglichkeiten:

- Die zu editierende Zeile kann mit einer Zeichenkette vorbelegt werden.
- Mit den Pfeiltasten läßt sich der Cursor links und rechts über diesen und neu eingegebenen Text bewegen.
- Mit „Backspace" wird der Cursor wird nach links bewegt, gleichzeitg wird das links stehende Zeichen gelöscht.
- Die Eingabe von „Del" löscht das Zeichen unter dem Cursor.
- „Home" läßt den Cursor an den Zeilenanfang springen, „End" ans Zeilenende.
- Durch „Ins" wird der Einfügemodus ein- oder ausgeschaltet.
- Die komplette Zeile wird durch „ESC" gelöscht, der Cursor springt an den Zeilenanfang.
- Mit „Carriage Return" schließlich wird die Eingabe abgeschlossen, die Zeichenkette kann von der übergeordneten Funktion weiter verarbeitet werden.

Weiterhin wird die Funktion abgeschlossen, wenn eine der Funktionstasten (auch mit Shift und Alt), die Cursor-up-, Cursor-down-, Page-up- oder Page-down-Tasten betätigt werden. Die Eingabe steht dem Anwender für eine Weiterverarbeitung zur Verfügung.

```
int editline(string, attribut, zeile, spalte,
                                max_laenge);
 char *string;
 unsigned char attribut;
 int zeile, spalte, max_laenge;
```

Eine (eventuell vorbelegte) Zeichenkette wird ab der gewünschten Position (Zeile und Spalte) mit dem angegebenen Attribut ausgegeben und kann editiert werden. Außerdem läßt sich die maximale Länge dieser Zeichenkette angeben. Sie sollte allerdings das Zeilenende nicht überschreiten: In diesem Fall wird die Länge auf das Zeilenende begrenzt. Ist die Länge mit Null angegeben, so wird sie intern auf Maximum = 80 Zeichen gesetzt. Die Eingabezeile wird vorher im angegebenen Format gelöscht. Da an die Funktion nur der Zeiger auf die Zeichenkette übergeben wird, kann nach Beendigung des Programms die neue Zeichenkette weiter verwendet werden. Der Tastencode, mit dem die editline-Funktion abgeschlossen wurde, wird an die übergeordnete Funktion zurückgeliefert. Die Definition der Tastencodes findet man in der Datei keyboard.h. Mit editline wird die aktuelle Cursorposition auf das Ende der Zeichenkette gelegt. Ist deren Länge gleich Null, so steht der Cursor auf der vorgegebenen Position.

Ein Beispiel

Um zu demonstrieren, wie die beschriebenen Funktionen benutzt werden können, sei auf das Programm in *Bild 5* verwiesen. In diesem Beispiel wird die Textausgabe, die Cursorpositionierung sowie das Auffangen der Funktionstasten aus der editline-Funktion demonstriert.
Das dazu gehörige Project-File muß wie folgt lauten:

```
demo (keyboard.h screen.h)
screen (keyboard.h screen.h)
editline (keyboard.h screen.h screen.c)
```

Die Demo wird mit Carriage Return abgeschlossen. Mit Betätigung der Funktionstasten erscheint auf dem Bildschirm eine entsprechende Meldung. Der Eingabemodus bleibt davon unbeeinflußt. □

Werner K. G. Münchheimer

Turbo-Pascal ruft 68008

Die 68008-Karte für den Apple nach [1] hat trotz anfänglicher Hardware-Probleme viele Liebhaber gefunden. Sie läßt sich mit wenigen Turbo-Pascal-Routinen als Coprozessor für CP/M-80 nutzen – oder als zweiter Masterprozessor wie im Falle des mc-68-ECB.

Der Wechsel vom Z80-Betriebssystem zum 68008 wird durch den 6502 des Apple als gemeinsamen Slave-Prozessor organisiert. Vor der ersten Benutzung oder nach Reset muß der 68008 erst initialisiert werden, dies geschieht mit INIT68K (*Bild 1*). Ein neues Preboot-Programm für DOS 3.3 (*Bild 2*) hilft über die bekannten Startprobleme beim Boot des CP/M hinweg. Das Programm CALL68K führt die bei jedem Wechsel notwendigen Berechnungen und Vorbesetzungen aus, es übernimmt als Parameter die Aufrufadresse der gewünschten 68008-Subroutine.

Anwendungen

Ein Fließkommapaket, wie in [2] beschrieben, eignet sich zur Beschleunigung umfangreicher Berechnungen mit bis zu 10 000 Fließkommaoperationen pro Sekunde. Gegenüber den in Grafikpaketen gebräuchlichen Anrufen von ROM-Routinen bringen 68000-Assembler-Routinen rund 5mal höhere Geschwindigkeit. Die Stärke der Karte wird allerdings erst deutlich bei Verwendung der 32-Bit-Integer-Befehle. Als Test kann das bekannte Apfelmännchen herhalten. In Turbo-Pascal dauert das Übersichtsbild etwa 37 Minuten, mit Iteration in 68000-Assembler 2 Minuten, 37 Sekunden und vollständig in 68000-Assembler nur 42 Sekunden. Der Hexdump (*Bild 3*) dieses Programms, mit DDT eingegeben, kann zum Test der Aufrufprozedur benutzt werden.

Literatur

[1] *Rojas, Darius, Drepper:* 68008-Karte für den Apple-II. mc 1985, Heft 9, Seite 46.

[2] *Lohninger, Hans:* Gleitkomma-Arithmetik für den 68000. mc 1985, Heft 2, Seite 58.

```
PROGRAM INIT68K; {Slot 2}
CONST SPPC68K: ARRAY   [1..8]  OF BYTE =
                 ($00,$00,$F3,$00,$00,$00,$F3,$00);
      CODE68K: ARRAY   [1..18] OF BYTE =
                 ($4E,$B8,$F3,$10,$11,$FC,$00,$C4,$F3,
                  $C8,$11,$C0,$B2,$00,$60,$F0,$4E,$75);
VAR   VEC_CPM: BYTE    ABSOLUTE $0000;
      VEC_BUF: ARRAY   [1..8] OF BYTE;
      MAIN68K: BYTE    ABSOLUTE $F300;
      CPU    : BYTE    ABSOLUTE $F3C8;
      SUB65  : INTEGER ABSOLUTE $F3D0;
      SLOT_80: INTEGER ABSOLUTE $F3DE;
BEGIN
   MOVE (VEC_CPM,VEC_BUF,8);
   MOVE (SPPC68K,VEC_CPM,8);
   MOVE (CODE68K,MAIN68K,18);
   MEM [$F307]:= CPU;
   SUB65:= $FF58; CPU:= $C2;
   MEM [SLOT_80]:= 0;
   MOVE (VEC_BUF,VEC_CPM,8);
END.
PROCEDURE CALL68K (ADDRESS: INTEGER); {Slot 2}
VAR SUB68K : INTEGER ABSOLUTE $F302;
    CPU    : BYTE    ABSOLUTE $F3C8;
    SUB65  : INTEGER ABSOLUTE $F3D0;
    SLOT_80: INTEGER ABSOLUTE $F3DE;
BEGIN
   SUB68K:= SWAP (ADDRESS); CPU:= $C2;
   SUB65:= $FF58; MEM [SLOT_80]:= 0;
END;
```

Bild 1. Turbo-Pascal-Routinen zur Initialisierung und zum Aufruf der 68008-Karte

```
10  TEXT : HOME : PRINT "CP/M BOO
    T MIT 68008 KARTE": PRINT : PRINT
    "INITIALISIERUNG:": PRINT
15 A$ = "1000:00 00 F3 00 00 00 F
    1 00 N 100:11 C0 B2 00 11 C0
     B2 00 4E F8 F3 00 N D7D2G"
20  FOR N = 1 TO  LEN (A$): POKE
    511 + N, ASC ( MID$ (A$,N,1)
    ) + 128: NEXT : CALL  - 144
25  INPUT "SLOT NR. DER 68008 KAR
    TE ? ";N: PRINT : POKE 258,1
    76 + N: POKE 262,176 + N: POKE
    256 * (192 + N),0
30  PRINT "LEGEN SIE DIE CP/M STA
    RTDISKETTE EIN": PRINT : INPUT
    "DRÜCKEN SIE <RETURN> ZUM BO
    OTEN";A$: PR# 6
```

Bild 2. Das Preboot-Programm für DOS 3.3

Bild 3. Dieses Testprogramm zeichnet in 42 Sekunden ein Apfelmännchen

```
F200- 31 FC D8 F3 F3 D0 11 C0
F208- B2 00 30 7C 00 23 34 7C
F210- 00 5F 36 7C 00 AA 49 F8
F218- F2 B2 4B F8 F2 BE 3E 3C
F220- 00 FF 34 3C 00 BF 32 3C
F228- 01 17 32 4B 36 02 38 02
F230- E8 0B 1A 34 30 00 E2 15
F238- 02 03 00 03 02 04 00 07
F240- D6 35 40 00 E1 4B 16 05
F248- 3C 43 30 08 36 09 38 0A
F250- 3A 03 DA 44 3C 03 9C 44
F258- CD C5 EC 46 9C 49 C9 C3
F260- EA 44 98 4A 3A 04 6A 02
F268- 44 45 36 06 6A 02 44 46
F270- DA 46 BA 47 5E C8 FF DA
F278- E2 48 65 14 36 01 48 C3
F280- 86 FC 00 07 68 02 48 43
F288- 38 03 48 43 07 F6 40 00
F290- 53 49 51 C9 FF B6 53 4A
F298- 51 CA FF 8C 31 FC 0C FD
F2A0- F3 D0 11 C0 B2 00 31 FC
F2A8- 99 F3 F3 D0 11 C0 B2 00
F2B0- 4E 75 00 00 00 00 50 50
F2B8- 50 50 A0 A0 A0 A0 30 34
F2C0- 38 3C 40 44 48 4C 00 00
```

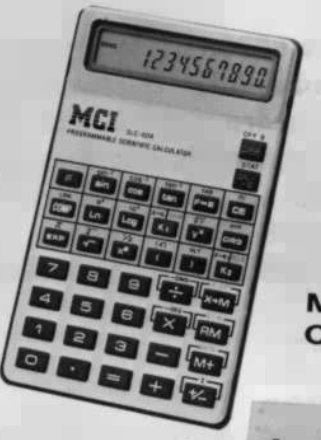
... KOMPLETTPREISE...SYSTEMPAKETE

KOMPLETTPREISE...SYSTEMPAKETE...1 JAHR GARANTIE

System Pakete für kluge Rechner

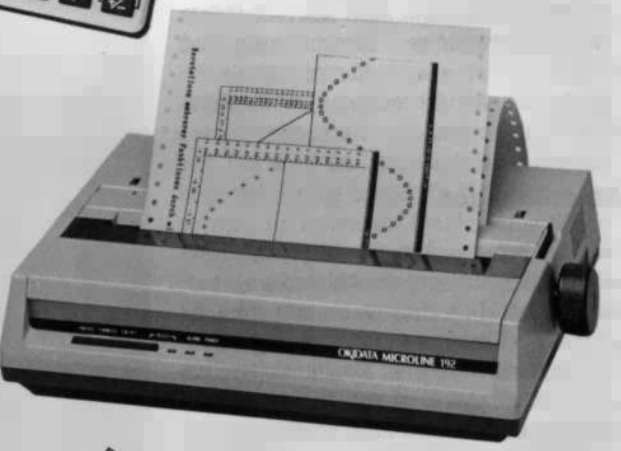
ab 1349,–

+



(+)

MCI Printer Plus **350,–**
OKI ML192 Elite **899,–**



System Paket 11

10 MHz

1349,–

- **MCI XT16SLC,** 640 K, 1 x 360 K, Clock, ser. par., 12" Monitor, MS-DOS, Tastatur
- **MCI** Programmierbarer Taschenrechner

opt. MCI 120 Zeichen Printer oder OKI ML 192 Elite (Mehrpreis siehe oben)

System Paket 21

10 MHz

1899,–

- **MCI XT16SLC,** 640 K, 1 x 360 K, Clock, ser. par., 12" Monitor, MS-DOS, Tastatur
- **MCI** 20MB Festplatte m. System formatiert
- **MCI** Programmierbarer Taschenrechner

opt. MCI 120 Zeichen Printer oder OKI ML 192 Elite (Mehrpreis siehe oben)

System Paket 31

12 MHz

2399,–

- **MCI AT4SLC,** 640 K, 1 x 1, 2 MB, Clock, ser. par., 12" Monitor, MS-DOS, Tastatur
- **MCI** Programmierbarer Taschenrechner

opt. MCI 120 Zeichen Printer oder OKI ML 192 Elite (Mehrpreis siehe oben)

System Paket 41

12 MHz

3199,–

- **MCI AT4SLC,** 640 K, 1 x 1, 2 MB, Clock ser. par., 12" Monitor, MS-DOS, Tastatur
- **MCI** 20MB Festplatte m. System formatiert
- **MCI** Programmierbarer Taschenrechner

opt. MCI 120 Zeichen Printer oder OKI ML 192 Elite (Mehrpreis siehe oben)

PREIS... QUALITÄT... 1 JAHR GARANTIE

KOMPATIBEL ... 24-STUNDEN-TEST ... LEISTUNG ...

MCI XT 16 SLC

alles drin!

Grundausstattung ohne Monitor

ab **899,–**

beinhaltet:

- voll IBM®XT kompatibel
- 8088 CPU + 8087 Sockel
- 8 XT Slots
- 256 KB freier Speicher
- 1 x 360 KB Floppy-Drive
- Color- oder Monochr. Grafikkarte (Hercules II komp. 720 x 348 P.)
- Deutsche Normtastatur MK 5111
- 150 W Schaltnetzteil
- Parallele Drucker-Schnittstelle

Erweiterungen für XT 16 SLC-Serie

2. Laufwerk 360 KB	249,–
Speichererweiterung auf 640 KByte	a. A.
Clock/Seriell-Karte	79,–
I/0 Plus II Karte	149,–
20 MB Festplatte mit XT-Controller	+ 589,–
30 MB Festplatte m. RLL XT-Contr.	+ 679,–
EGA-Set statt monochr. Karte	+ 1149,–
Opt. Maus mit Tablet	119,–
Professional Multifunktions-Tastatur MK 6000	+ 100,–
MS-DOS 3.2 + GW-Basic	+ 149,–
9" TTL Monitor grün	+ 150,–
12" Monitor grün od. bern.	+ 229,–
14" TTL Monitor grün, bern. od. weiß	+ 279,–

MCI AT 4 SLC

alles drin!

Grundausstattung ohne Monitor

ab **1899,–**

beinhaltet:

- voll IBM® AT kompatibel
- 80286 CPU + 80287 Sockel
- 6 AT + 2 XT Slots
- 8 und **12 MHz** umschaltbar
- 512 KB freier Speicher
- 1 x 1,2 MB/360 KB Laufwerk
- Color- oder Monochr. Grafikkarte (Hercules II komp. 720 x 348 P.)
- Parallele Drucker-Schnittstelle
- Batteriegep. Echtzeituhr/Kalender
- Kapazitive deutsche Normtastatur

Erweiterungen für AT 4 SLC-Serie

2. Laufwerk 360 KB	299,–
20 MB Festplatte mit AT-Controller	889,–
30 MB Festplatte m. RLL AT-Contr.	+ 999,–
Seriell-Karte	79,–
I/0 Plus II Karte	149,–
EGA-Set statt monochr. Karte	+ 1149,–
MS-DOS 3.2 + GW-Basic	+ 149,–
Professional Multifunktions-Tastatur MK 6000	+ 100,–
9" TTL Monitor grün	+ 150,–
12" Monitor grün od. bern.	+ 229,–
14" TTL Monitor grün, bern./weiß	+ 279,–

PRINTER

MCI *Personal Computer Graphics Printer Plus*

- **voll kompatibel zum** *IBM Personal Computer Graphics Printer*
- **120 Zeichen/sec.**

399,–

OKI MICROLINE ML 192 Elite

- 9 Nadel Matrixdrucker
- Druckgeschwindigkeit 200 Z./sec.
- 40 Zeichen/sec. NLQ
- Druckpuffer 8 KB
- IBM Kompatibel

899,–

EGA

Hochauflösender EGA-Monitor

- Auflösung 320 x 200 (CGA Mode) 640 x 350 (EGA Mode)

EGA-Karte 249,–

999,–

TELEFON-HOTLINE-PREISE
MCI Telefonansage (20 sec.)

	(0 22 02)
Festplatten & Controller	10 81 40
Grafikkarten	10 81 41
Monitore	10 81 42
Schnittstellenkarten	10 81 43
Drucker	10 81 44
Aktuelle Neuigkeiten	10 81 45

MCI

5060 Bergisch Gladbach 2
Bensberger Straße 252
Tel.-Nr.: 02202/1080
Fax: 02202/31009 · Telex: 8873518

12 Monate Garantie auf alle Geräte. Nach der Pang Vo. v. 14. 3. 85 sind wir bei Angeboten gegenüber dem Endverbraucher zur Angabe der **Preise incl. MwSt.** verpflichtet. Für Druckfehler wird nicht gehaftet. **Preise gültig ab 1. 4. 88.** Lieferzeit und Lieferbedingungen auf Anfrage. Änderungen, die technischen Verbesserungen dienen, vorbehalten. Zwischenverkauf vorbehalten.
MCI MICRO COMPUTER INSTRUMENTS GMBH eingetragen AG Bergisch Gladbach · HRB 2575. Herstellung und Vertrieb von Mikrocomputern. 5060 Bergisch Gladbach 2 · Bensberger Straße 252

Karl-Heinz Schulz

CGA-Emulation auf der Hercules-Karte

Viele Programme setzen eine CGA-Karte im System voraus. Alle Grafikbefehle, die GW-Basic kennt, funktionieren nicht auf einer Hercules-Karte. Bevor man sich eine Farbgrafikkarte und einen teuren Farbmonitor zulegt, sollte man prüfen, ob nicht bereits ein CGA-Emulator ausreicht.

Grafikkarten nach dem Hercules- und dem CGA-Standard arbeiten mit dem Video-Controller 8645. Dennoch gibt es zwischen beiden Karten beträchtliche Unterschiede. Jeder der schon einmal versucht hat, CGA-Programme auf der Hercules-Karte laufen zu lassen, kennt die Probleme. Offensichtlich gibt es Unterschiede, die in der Karte selbst und nicht beim Video-Controller liegen. Betrachtet man die höchstmögliche Auflösung der beiden Karten, fällt sofort auf, daß der CGA-Wert (640 x 200) innerhalb des Hercules-Wertes (720 x 348) liegt. Damit sollte es also möglich sein, die CGA-Karte auf der Hercules-Karte zu emulieren, wenn man von der Farbdarstellung absieht. Die einfachste Möglichkeit einer Emulation wäre, den Video-Controller entsprechend zu programmieren. Jedoch reicht es nicht aus, einfach die CGA-Werte zu nehmen, da die Werte für das HSync- und das VSync-Signal bei beiden Karten unterschiedlich sind. Deswegen ergeben sich auch bei einigen anderen Registern unterschiedliche Werte. Die Unterschiede der Programmierung der Controller-Register sind in der *Tabelle 1* dargestellt. Einige Berechnungen führen zu den Werten der Param-n-Tabellen im Programm.

Das Programm CGAHERC

Bei allen IBM-kompatiblen PCs steht zur Bildschirmausgabe ein Software-Interrupt – der Interrupt 10H – dessen Interrupt-Service-Routine im BIOS steht, zur Verfügung. Mit dem Interrupt 10H kann man verschiedene Funktionen aufrufen. Dazu muß man vor dem Auslösen des Interrupts die Nummer der gewünschten Funktion ins AH-Register schreiben. Einen Überblick über die wichtigsten Funktionen des Interrupts gibt die *Tabelle 2*.
Um den Video-Interrupt zur Emulation der CGA-Karte zu verwenden, muß man dafür sorgen, daß nicht die Interrupt-Service-Routine im BIOS angesprungen wird, sondern die Emulationsroutinen des Programms CGAHERC. Das Verfahren wird einleuchtend, wenn man die Interrupt-Behandlung der Intel-Prozessoren kennt.
Bei ihnen ist jedem der 265 möglichen Interrupts ein Eintrag in der Interrupt-Vektor-Tabelle zugeordnet. Man erhält die Adresse des jeweiligen Eintrags, indem man die Nummer des Interrupts mit vier multipliziert. Die Tabelle belegt die Adressen 000H...3FFH. Ein Eintrag ist vier Byte lang und stellt einen Zeiger auf die zugehörige Interrupt-Service-Routine im Offset-Segment-Format dar. Der größte Teil des Programms CGAHERC bleibt resident im Speicher. Es verändert die Vektoren der Interrupts 10H, 11H (Hardware-Ausstattung) und 42H (Zeiger auf EGA-BIOS). Jeder Aufruf des Video-Interrupts 10H wird zur Routine NEWINT10 umgeleitet. Hier erfolgt zunächst der Test, ob es sich um eine Kennungsabfrage von CGAHERC handelt. Falls das zutrifft, liefert sie den Parameter INSTALLED zurück. Sonst erfolgt die Abfrage, ob es sich um die Funktion 0 des Interrupts 10H (Text- und Grafikmodusauswahl) handelt. Ist dies der Fall, springt sie zur Modus-0-Routine. In allen anderen Fällen und als erstes in der Modus-0-Routine wird getestet, ob die Emulation aktiv ist. Ist sie nicht aktiv, wird der Aufruf zur alten BIOS-Service-Routine umgeleitet. Bei der Modus-0-Routine wird vorher noch getestet, ob es sich nicht um eine Aktivierung von CGAHERC (HercTxt) handelt und entsprechend verfahren.

Programmstruktur

Das Programm ist in die folgenden Bereiche aufgeteilt:
– Installationsteil (K_Rqst....CSEG ENDS)
– Emulationsteil (NewInt10...K_Rqst)
– Residenter Datenbereich (vor NewInt10)

Zunächst werden im Listing (*Bild 1*) einige Konstanten und Adressen deklariert. Danach wird der wichtigste Teil des BIOS-Daten-Segments definiert. Der Assembler erzeugt bei diesen Anweisungen keinen Code, sondern er merkt sich nur die Offset-Adressen der Daten im Daten-Segment. Konstanten werden während der Assemblierung durch ihren entsprechenden Wert ersetzt. Für das Programm ist der BIOS-Datenbereich, der bei Adresse 400H beginnt, von Bedeutung. Einige Adressen des BIOS-Datenbereichs (0040:0049H...0040:0066H) werden vom Programm angesprochen.

Es folgen Assembler-Direktiven zum Code-Segment. Aus ihnen entnimmt der Kenner, daß es sich um ein COM-Programm handelt, da die Assume-Direktive sowohl dem CS- als auch dem DS-Register das gleiche logische Segment (hier CSEG) zuweist und kein Stacksegment angelegt wird.

Nur im NEAR-Segment Set_Bios des Programms gilt ein anderes Daten-Segment. Es reicht aber nicht aus, nur die Assume-Direktive zu ändern, da sie nur eine assemblerinterne Auswirkung hat. Damit tatsächlich auf das gewünschte Segment zugegriffen werden kann, muß dem Register DS über AX (oder über den Stack) der Segmentwert zugewiesen werden. Da es sich bei dem Programm um eine COM-Datei handelt, ist beim Linken die Stack-Fehlermeldung zu ignorieren. Anschließend ruft man EXE2BIN wie folgt auf:

EXE2BIN CGAHERC.EXE CGAHERC.COM
Die Objekt- und die EXE-Datei wird nicht mehr gebraucht und darf gelöscht werden. CGAHERC.COM ist das lauffähige Emulationsprogramm.

Die Installations-Routine

Die Routine Do_Init prüft zunächst mit der Funktion K_Rqst, ob die Emulation bereits installiert ist. Die Videofunktion KENNUNG existiert beim Interrupt 10H nicht

Tabelle 1: So sind die Register des Video-Controllers 6845 bei der CGA und der Hercules-Karte programmiert

Register	Bit	Hercules-Textmodus MDA	CGA-Textmodus 80×25	CGA-Textmodus 40×25	CGA Grafikmodus 640×200	Hercules-Grafikmodus 720×348	CGAHERC Emulation	Bedeutung
0	0...7	97	113	56	56	53	53	Zahl der Textzeichen pro Zeile –1 für Hsync
1	0...7	80	80	40	40	45	40	Zahl der tatsächlich angezeigten Zeichen/ Zeile
2	0...7	82	90	45	45	46	46	Rasterzeilenposition des HSync-Signals
3	0...3	15	10	10	10	7	6	Länge d. HSync-Signals
	4...7	0	0	0	0	0	0	(Länge d. HSync-Signals) (Dauer immer 16 Scan)
4	0...6	25	31	31	127	91	124	Zahl der Textzeilen –1
5	0...4	6	6	6	6	2	15	Feineinstellung für Register 4 in Rasterzeilen
6	0...6	25	25	25	100	87	100	Zahl der tatsächlich angezeigten Zeilen
7	0...6	25	28	28	112	87	115	Zeichenposition des VSync-Signals –1
8	0...1	2	2	2	2	2	2	Interlace (ein: 1,3; aus: 0,2)
9	0...4	13	7	7	1	3	2	Höhe eines Zeichens in Rasterzeilen –1
10	0...4	11	6	6	6	0	0	Erste Rasterzeile des Cursors (maximal 12)
	5...6	0	0	0	0	0	0	Spezielle Cursormodi (beim IBM nicht normgerecht)
11	0...3	12	7	7	7	0	0	Letzte Rasterzeile des Cursors (maximal 12)
12	0...5	–	–	–	–	–	–	Bit 8–13 der Startadresse d. Bildschirmspeichers
	6...7	0	0	0	0	0	0	---
13	0...7	–	–	–	–	0	0	Bit 0...7 der Startadresse des Bildschirmspeichers
14	0–7	–	–	–	–	0	0	Bit 8–13 der Adresse des Cursors (maximal 1999)
	6...7	0	0	0	0	0	0	---
15	0...7	–	–	–	–	0	0	Bit 0–7 der Adresse des Cursors (maximal 1999)
16	0...5	–	–	–	–	0	0	Bit 8–13 der Schirmadresse des Lichtgriffelimpulses
	6...7	0	0	0	0	0	0	---
17	0...7	–	–	–	–	–	–	Bit 0–7 der Schirmadresse des Lichtgriffelimpulses

Alle Werte sind dezimal

Tabelle 2: Die wichtigsten Video-Funktionen des Interrupts 10H

Funktion	Bedeutung
0	Text- bzw. Grafikmodusauswahl
1	Cursorgröße wählen
2	Cursor positionieren
3	Cursorposition holen
4	Lichtgriffelposition holen
5	Aktive Bildschirmseite setzen
6	Bildschirm aufwärts scrollen (auch für Clear Screen)
7	Bildschirm abwärts scrollen
8	Zeichen und Attribut unter Cursor lesen
9	Zeichen und Attribut an Cursorposition schreiben
10	Zeichen ohne Attribut an Cursorposition schreiben
11	Farbpalette oder Farbe auswählen
12	Grafikpunkt setzen
13	Grafikpunkt lesen
14	Zeichen an Cursorposition schreiben und Cursor weiterbewegen (TTY-Funktion)
15	Bildschirmstatus ermittein

Tabelle 3: Die Geräteausstattung eines PCs steht in Adresse 0040:0010

Bits	Bedeutung
15, 14	Anzahl der angeschlossenen Drucker
13	–
12	Game-Karte angeschlossen
11, 10, 9	Anzahl der angeschlossenen RS232-Karten
8	–
7, 6	Anzahl Diskettenlaufwerke 00=1 01=2 10=3 11=4
5, 4	Video-Modus 00 – – 01 – 40×25 CGA 10 – 80×25 CGA 11 – Monochromadapter
3, 2	Grundausstattung RAM
1	–
0	Diskettenlaufw. vorhanden, wenn ja, dann Bit 7+6

und wird lediglich von CGAHERC zur Verfügung gestellt und liefert den Parameter INSTALLED zurück. Ist das Programm noch nicht installiert, werden jetzt die Interruptvektoren VIDEO AUSSTATTUNG und EGA-BIOS (10H, 11H, 42H) über dieses Programm geleitet und die Operation im KENNFLAG vermerkt. Anschließend folgt die Parameterüberprüfung des Aufrufes.
Danach – außer beim Booten – wird das Programm mit einer von zwei Möglichkeiten verlassen: Ist der Wert im KENNFLAG gleich false, bleibt das Programm resident (Interrupt 27H), sonst wird es normal über den Interrupt 20H verlassen.

Schirmauswahl

Ab dem Label ScrSel0 sind nicht nur die Standardfunktionen des Video-Interrupts vorhanden, sondern auch die erweiterten Funktionen 'Emulation ein', 'Emulation aus', 'Olivetti-Modus 640x400' und 'Herculesgrafikmodus für Seite 2'. Je nach Modus – Text oder Grafik – wird entsprechend zu den beiden Routinen P_TEXT und P_GRAF verzweigt.
Die Routine P_TEXT ruft die Routine Set_Mode auf, welche die Hercules-Karte vom Grafikmodus auf den Textmodus umschaltet. Sicherheitshalber setzt sie das Equipment-Flag im BIOS-Daten-Segment (Offset 10H) zurück. Das Flag entspricht dem

Wert, den der Interrupt 11H (Geräteaustattung) liefert. Die Bedeutung der einzelnen Bits des Inhalts unter Adresse 0040:0010H geht aus *Tabelle 3* hervor.
Die Routine P_GRAF dagegen ist weit aufwendiger gestaltet. Sie löscht zunächst den Bildwiederholspeicher im Segment B800H, welcher der CGA-Karte sowie der zweiten Seite der Hercules-Karte im Grafikmodus zugeordnet ist. Bevor sie die alte Interrupt-Service-Routine im BIOS aufruft, setzt sie die BIOS-Parameter FlagSet, Set_Mode und Set_Col. Die alte BIOS-Routine setzt weitere Parameter. Erst dann programmiert die Prozedur SET_CRT den CRT-Controller mit den Werten des Feldes PARAM5.
Es folgt ein Aufruf der Prozedur Screen_U, die den Bildschirm dunkel tastet (Scr_Off), eine Warteschleife (Lp_Lag) durchführt, einen neuen Wert in das BIOS-Daten-Segment schreibt und den Bildschirm wieder hell tastet (Scr_On mit SetCRTMod).

Sonstige Video-Routinen

Da die Hercules-Karte keine Farbattribute kennt, müssen die entsprechenen Routinen ersetzt (WritPair, DoColor, Rollen) oder umgewandelt werden. Für die Programme, die den Video-Modus über den Interrupt 10H abfragen und den Monochrom-Modus nicht akzeptieren, wird der Modus 7 nicht zurückgeliefert (Modi_Tst mit ScrGet). Ebenso ersetzt CGAHERC die BIOS-Routinen 'Punkt setzen' und 'Punkt holen' mit den Routinen für die folgenden Betriebsarten:
- CGA-Modus 640x200 mit einem Zeilen-Interleave von zwei
- Olivetti-Modus 640x400 mit einem Zeilen-Interleave von vier
- Hercules-Modus 720x348 mit einem Zeilen-Interleave von vier.

Bis auf die Zeilenlänge (80 gegenüber 90 Bytes) ist die 'Punkt setzen'- Routine des Olivetti-Modus mit der entsprechenden Routine der Hercules-Karte identisch. Für die Routinen 'Punkt holen' gilt das Gleiche. Das Register BP enthält im Programm den entsprechenden Wert.

Equipment-Flag-Wort

Außer dem Video-Interrupt wird auch der Interrupt 11H umgeleitet. Das dient zum Täuschen von hartnäckigeren Programmen, die den aktuellen Schirmmodus testen. Solche Programme sollte man erst aufrufen, wenn die CGA-Emulation bereits aktiv ist, damit auch die Interrupt-11H-Abfrage den richtigen Wert liefert. Manche Programme schalten in den Textmodus und fragen dann

```
.8086
TITLE   CGAHERC Farbgrafikemulation
NAME    CGAHERC
SUBTTL  Vorlauf
PAGE    60,130
;
COMMENT $ BEGIN Emulation  für (IBM) PC/AT
        der Farb-Grafik-Karte (CGA) auf der Hercules-Grafik-Karte (HGC)
Hinweise 'S. Bemerkung ##' mit flgd. Bedeutung:
    10:   Einige Programme geben bei dieser Aktivierung im Textmodus
          keine Texte aus. Ist der Aufruf aktiviert, so konnen flgd.
          Parameter helfen, die Emulation zu optimieren:
    11:   Bei einigen Programmen   JZ -> JMP setzen
    12:   Behauptet CGA ist da (INT 11H)
    13:   Behauptet CGA ist da (equipment-Flag)
    30:   Ein paar Programme arbeiten mit den flgd. Angaben bei der
          Textausgabe besser. Man konnte die meisten zur besseren
          Textausgabe bekommen, wenn man die Textausgabe vollstandig
          selber verwalten wurde!
    31:   Textmodi der CGA-Karte werden als Grafik emuliert
    32:   Wenn 31 zutrifft, darf CopyPage nicht aktiviert werden!
END $
;----------------------------------------
;- Konstantendefinition                 -
;----------------------------------------
false       equ  0000H                  ; Byte 00H oder Wort 0000H
true        equ  not false              ; Byte FFH oder Wort FFFFH
installed   equ  0DD00H                 ; Return-Wert VIDEO(Kennung)
mono        equ  00101001B              ; (29H)
Z40         equ  40D                    ; 40 Zeichen/Zeile
Z80         equ  80D                    ; 80 Zeichen/Zeile
Space       equ  ' '
Ctrl_C      equ  03H
Scr_On_Flag equ  00001000B
;----------------------------------------
;- Interrupts                           -
;----------------------------------------
Video       equ  10H                    ; Bildschirmservice 00..0F :
   ScrSel   equ  00H            ; Modusauswahl durch AH=0,AL=0..7
    MonoMod equ  0007H          ;   Monochrom-Modus
    OliMod  equ  0040H          ;   CGAHERC 640x400
    HercGrf equ  00F0H          ;   CGAHERC 720x348
    HercTxt equ  00F1H          ;   CGAHERC monochrom
    NoEmul  equ  00FFH          ;   CGAHERC keine Emulation
   CurSize  equ  01H            ; Cursorform in DH,DL
   CurPos   equ  02H            ; Cursor positionieren mit DH=Zeile, DL=Spalte
   CurGet   equ  03H            ; Cursorposition holen
   ScrollUp equ  06H            ; Schirm aufwarts rollen
   ScrollDw equ  07H            ; Schirm abwarts rollen
   WritPair equ  09H            ; schreibe Zeichen in AL mit Attribut in BL
   WritChar equ  0AH            ; schreibe Zeichen in AL ohne Attribut
   DoColor  equ  0BH            ; Grafikfarbe an Cursorposition
   PntSet   equ  0CH            ; Grafikpunkt Farbe setzen
   PntGet   equ  0DH            ; Grafikpunkt Farbe holen
   PntRead  equ  0DH            ; Grafikpunkt lesen
   WritTTY  equ  0EH            ; schreibe Zeichen und Cursor
   ScrGet   equ  0FH            ; Modus holen: AL=Modus, AH=Spalten
    kennung equ  0D000H         ;   CGAHERC-Kennungsabfrage
Hardware    equ  11H                    ; Gerateausstattung
Tast_Ein    equ  16H                    ; Tastatureingabe
   GetChar  equ  00h            ; warte auf Tastendruck und liefere Zeichen
Boot_Int    equ  19H                    ; Warmboot
Pend        equ  20H                    ; Programm beenden
MsDos       equ  21H                    ; Funktionsaufrufe
   StrgOut  equ  09H            ; Stringausgabe bis '$'
   SetVec   equ  25H            ; Interrupt-Vektoradresse setzen aus DS:DX
   GetVec   equ  35H            ; Interrupt-Vektoradresse holen  in  ES:BX
   FreMem   equ  49H            ; Speicherplatz-Freigabe
Terminate   equ  27H                    ; wie PEND, aber resident bleiben
EGA_BIOS    equ  42H                    ; Zeiger auf Eingang EGA-BIOS
;----------------------------------------
;- Port-Adressen                        -
;----------------------------------------
indx_reg    equ  3B4H                   ; 6845 Adress-(Index-)Register Hercules
  cga_index equ  3D4H                   ; 6845 Adress-(Index-)Register CGA
data_reg    equ  3B5H                   ; 6845 Data Register  Hercules
  cga_data  equ  3D5H                   ; 6845 Data Register  CGA
ctrl_reg    equ  3B8H                   ; Display Mode Control Port
status_reg  equ  3BAH                   ; Display Status Port
CRT_Config  equ  3BFH                   ; Configuration Data Switch
;----------------------------------------
;- CODESEGMENTPOINTER                   -
;- DATENSEGMENTE sowie Offsetadressen   -
;-            des BIOS-Datensegmentes   -
;----------------------------------------
ESegA       equ  002CH                  ; Environment-Segment-Adress
DOS_Puffer  equ  000B0H                 ; im aktuellen Kode-Segment
Scrn0_Base  equ  0B000H                 ; Bildwiederholspeicher-Segment
Scrn1_Base  equ  (Scrn0_Base+800H)      ; CGA-Segment, Hercules Seite 2
Bios_Data       SEGMENT AT 40H          ; Offsetadressen in HEX
  RS232_BASE     DW  4 dup(?)           ; 00H
```

Bild 1. Das Emulations-Programm ist in Assembler geschrieben

```
  PRINTER_BASE  DW  4 dup(?)
  equIP_Flag    DW  ?                   ; 10H
  MFG_Flag      DB  ?
  Memory_Size   DW  ?                   ; 13H
  MFG_err_Flag  DB  ?
                DB  ?
  KB_FLAG       DB  2 dup(?)
  Alt_Input     DB  ?
  Buffer_Head   DW  2 dup(?)            ; 1AH
  KB_Buffer     DW 16 DUP(?)            ; 1EH
  Disk_Parms    DB 11 DUP(?)            ; 3EH
  CRT_Mode      DB  ?                   ; 49H
  CRT_Cols      DW  ?
  CRT_Len       DW  ?
  CRT_Start     DW  ?
  Cursor_Posn   DW  8 DUP(?)
  Cursor_Mode   DW  ?                   ; 60H
  Active_Page   DB  ?
  ADDR_6845     DW  ?
  CRT_Mode_Set  DB  ?                   ; 65H
  CRT_Palette   DB  ?
  IO_ROM        DW  2 dup(?)            ; 67H
  Intr_Flag     DB  ?                   ; 6BH
Bios_Data       ENDS
;=======================================
;= Assembler: Segment- und             =
;=            Organisationsbefehle     =
;=======================================
CSEG            SEGMENT 'CODE'
                ASSUME CS:CSEG, DS:CSEG, SS:NOTHING
ORG             100H
;
START:   JMP      Beginn
SUBTTL   Emulation
PAGE +
;=======================================
;Residenter Datenbereich dieses Treibers
;=======================================
Comment $ Grober Registeraufbau des Bildschirmcontrollers 6845
  0     Gesamtanzahl Zeichen per Zeile inklusive SYNC - 1
  1     Anzahl der sichtbaren Zeichen per Zeile
  2     Position des ersten Zeichens während SYNC -1
  3     Anzahl Zeichen während SYNC -1
  4     Anzahl der Zeilen -1, inklusive der Zeile während vertical retrace
  5     Anzahl Scanzeilen (Zeilenanteil) zusätzlich zu Register 4
  6     Anzahl der sichtbaren Zeilen
  7     Zeilennummer zum Beginn des retrace -1 (Dauert immer 16 Scans)
  8     Interlace: Hat bei der Herculeskarte IMMER den Wert 2
  9     Anzahl der Scans per Zeile -1
 10,11  Erste und Letzte Cursor-Scan-Zeile
 12,13  Immer 0
 14,15  Cursorgröße, in der Emulation immer 0

! :      R1# "9"   = 720          R6#(R9+1)  = 348
Tabellen innerhalb Comment:
param1          DB 34H, 28H, 2AH, 47H, 69H      ; Grob-Emulation
                DB 00H, 64H, 65H, 02H, 03H
                DB 00H, 00H, 00H, 00H, 00H, 00H
param2          DB 33H, 28H, 2BH, 07H, 67H      ; Mittlere Auflösung
                DB 0AH, 64H, 65H, 02H, 03H
                DB 00H, 00H, 00H, 00H, 00H, 00H
param3          DB 35H, 28H, 2BH, 0AH, 7BH      ; Fein-Emulation 1
                DB 06H, 64H, 6CH, 02H, 02H
                DB 00H, 00H, 00H, 00H, 00H, 00H
param4_1        DB 36H, 28H, 2EH, 07H, 79H      ; Fein-Emulation 2
                DB 0DH, 64H, 73H, 02H, 02H
                DB 00H, 00H, 00H, 00H, 00H, 00H
param4_2        DB 36H, 28H, 2EH, 06H, 7EH      ; Fein-Emulation 3
                DB 0DH, 64H, 73H, 02H, 02H
                DB 00H, 00H, 00H, 00H, 00H, 00H
param4_4        DB 36H, 28H, 2DH, 08H, 7FH      ; Fein-Emulation 4
                DB 1FH, 64H, 72H, 02H, 01H
                DB 00H, 00H, 00H, 00H, 00H, 00H
End $
param5          DB 36H, 28H, 2EH, 06H, 7CH      ; Spezialemulation
                DB 0FH, 64H, 76H, 02H, 02H
                DB 00H, 00H, 00H, 00H, 00H, 00H
param_Herc      DB 36H, 2DH, 2EH, 07H, 5BH      ; Orig. Hercules-Grafik
                DB 02H, 57H, 57H, 02H, 03H
                DB 00H, 00H, 00H, 00H, 00H, 00H
param_Mono      DB 61H, 50H, 52H, 0FH, 19H      ; Hercules-Textmodus
                DB 06H, 19H, 19H, 02H, 0DH
                DB 0BH, 0CH, 00H, 00H, 00H, 00H
param_ptr       DW param5                       ; durch ScrSel# gesetzt
ES_Stack        DW 0000H                        ; Zwischenspeicher für ES
COMMENT $ BEGINN
          Scr_Mode_Set ist Kopie von CRT_Mode_Set im BIOS-Datensegment
          CGA_Mode_Set ist Kennzeichen für CGA-Modi 0-6 und ist mit
          Scr_Modi     speziell für die Modi 0-3 vorgesehen.
                       (Derzeit noch nicht im Einsatz!)
END $
Scr_Mode_Set    DB mono                         ; mono/graf/mono
CGA_Mode_Set    DB mono                         ; 0-3 /graf/mono
Scr_Modi        DB 2                            ; 0-3 /4-6 /  2
Scr_Cols        DB 40H
Oli_Mod         DB LOW false
Hercules_Mod    DB LOW true
Emulation       DB LOW true
COPY$:          DB  (C) 13.01.1988 by KHS Koln, CGA - Emulation  V4,0$'
EVEN
Old_Int10       DD 0000H
Old_Int11       DD 0000H
Old_Int16       DD 0000H
Old_Int42       DD 0000H
;
NewInt10 PROC   NEAR
         PUSH   ES
         PUSH   DS
         PUSH   SI
         PUSH   DI
         PUSH   AX
         MOV    AX,CS
         MOV    DS,AX                   ; DS:=CS
         MOV    [ES_Stack],ES           ; für eigene Zwecke
         MOV    ES,AX                   ; ES:=CS
         POP    AX
         CMP    AX,Kennung              ; IF     Kennung request
         JE     Fn_Kennq                ; THEN   Kennung-Funktion
         OR     AH,AH                   ; IF     Schirmmodusauswahl
         JZ     ScrSel0                 ; THEN   neue Routine (cgaherc)
         CMP    [Emulation],false       ; IF     keine Emulation
         JE     DoOldInt                ; THEN   alter INT10-Aufruf
         CMP    [Scr_Mode_Set],mono     ; Im Mono-Modus
         JE     DoOldInt                ;        alte Routine (BIOS-INT10)
Pair_Tst:CMP    AH,WritPair             ; IF NOT Schreiben mit Attribut
         JNE    Roll_Tst                ; THEN   nachster Test
         MOV    AH,WritChar             ; ELSE   Herculeskarte hat kein
                                        ; Attribut!
```

```
        JMP     SHORT DoOldInt
Roll_Tst:CMP    AH,ScrollUp             ; Rollen geschieht mit Attribut!
        JB      Clor_Tst
        CMP     AH,ScrollDw
        JA      Clor_Tst
        MOV     BH,00                   ; (nicht korrekt, aber besser)
        JMP     SHORT DoOldInt
Clor_Tst:CMP    AH,DoColor              ; Funktion FARBE SETZEN wird
        JE      RetInt                  ;          ersatzlos gestrichen
Modi_Tst:CMP    AH,ScrGet               ; Modus 7 ist ausgeblendet!
        JNE     Punkt_S
        CALL    FlagSet
        MOV     AL,[Scr_Modi]           ; Return:(1-6)
        MOV     AH,[Scr_Cols]           ; Return:Spalten(40/80)
        JMP     SHORT RetInt
Punkt_S: CMP    AH,PntSet               ; IF NOT Punkt setzen
        JNE     Punkt_H                 ; THEN   WEITER
        JMP     PNT_JMP                 ; ELSE   Sprungverteiler
;
Punkt_H: CMP    AH,PntGet               ; IF NOT Punkt holen
        JNE     _Tst                    ; THEN   WEITER
        JMP     Pnt_Hol                 ; ELSE   Sprungverteiler
_Tst:
DoOldInt:PUSHF                          ; - wegen RETI nachste Zeile
        CALL    DWord Ptr [Old_Int10]
        JMP     SHORT RetInt
Fn_Kenng:MOV    AX,installed            ;        say installed
RetInt:  POP    DI                      ; NewINT-10H - Return
        POP     SI
        POP     DS
        POP     ES
        IRET
;-----------------------------------------------
;- Schirmauswahl                               -
;-----------------------------------------------
ScrSel0: CMP    AX,NoEmul
        JNE     MonoTst
        MOV     [Emulation],false
        MOV     AX,MonoMod
;
MonoTst: CMP    AX,HercTxt
        JNE     HercTst
        MOV     [Emulation],true
        MOV     AX,MonoMod
;
HercTst: CMP    [Emulation],false       ; IF     keine Emulation
        JE      DoOldInt                ; THEN   alter INT10-Aufruf
        MOV     [Oli_Mod],false
        MOV     [Hercules_Mod],false
        CMP     AX,HercGrf              ; IF Hercules-Modus
        JNE     OliTst
        MOV     [Hercules_Mod],true
        MOV     [Emulation],true        ; wegen Punkt setzen
        MOV     [Scr_Cols],Z80
        MOV     [param_ptr],offset param_Herc
        JMP     P_GRAF
;
OliTst:  TEST   AX,OliMod               ; IF Olivetti 600x400 Punkte
        JZ      ScrCont
        MOV     [Oli_Mod],true
        MOV     AL,6
;
ScrCont: CMP    AL,7                    ; > 7 ?
        JA      RetInt
        MOV     [Scr_Modi],AL           ; für Return-Wert FN(ScrGet)
```

```
COMMENT $ Siehe Bemerkung 30; Aktivieren: Kommentar/-END entfernen
;         Dann auch Bemerkung 11 beachten 'JZ -> JMP'.
;         Mit ANSI.SYS funktioniert es tadellos; beim Booten wird
;         ANSI jedoch gelöscht. Abhilfe schafft hier nur eine eigene
;         Verwaltung nach Art von ANSI.SYS für CONSOLENAUSGABE
        cmp     al,3                    ; S. Bemerkung 31
        ja      scrsel1
        cmp     al,1
        mov     al,4
        ja      scrsel1
        mov     al,6
END $
        CMP     AL,00H                  ; 40x25   (B/W)
        JNZ     ScrSel1
        JMP     P_TEXT
;
ScrSel1: CMP    AL,01H                  ; 40x25   (farbig)
        JNZ     ScrSel2
        JMP     P_TEXT
;
ScrSel2: CMP    AL,02H                  ; 80x25   (S/W)
        JNZ     ScrSel3
        JMP     P_TEXT
;
ScrSel3: CMP    AL,03H                  ; 80x25   (farbig)
        JNZ     ScrSel4
        JMP     P_TEXT
;
ScrSel4: CMP    AL,04H                  ; 320x200 (S/W)
        JNZ     ScrSel5
        MOV     [Scr_Cols],Z40
        MOV     [param_ptr],offset param5 .
        JMP     P_GRAF
;
ScrSel5: CMP    AL,05H                  ; 320x200 (farbig)
        JNZ     ScrSel6
        MOV     [Scr_Cols],Z40
        MOV     [param_ptr],offset param5
        JMP     P_GRAF
;
ScrSel6: CMP    AL,06H                  ; 640x200/400 (S/W)
        JNZ     ScrSel7
        MOV     [Scr_Cols],Z80
        MOV     [param_ptr],offset param5
P_GRAF:  CALL   ClrScr
        CALL    Set_Bios
        SUB     AH,AH                   ; Funktion 0, INT 10H
        MOV     ES,[ES_Stack]           ; Original ES-Wert
        PUSHF                           ; - wegen RETI nachste Zeile
        CALL    DWord Ptr [Old_Int10]
        MOV     AH,10010011B
        MOV     AL,BYTE PTR [Scr_Mode_Set]
        AND     AL,00001100B
        OR      AH,AL                   ; 1001..11B
        MOV     [Scr_Mode_Set],AH
        MOV     [CGA_Mode_Set],AH
        AND     AH,11110111B
        CALL    Set_Crt
        CALL    Screen_U
        JMP     RetInt
;
ScrSel7:                                ; 80x25 Textmodus monochrom
        MOV     [CGA_Mode_Set],mono
        MOV     [Scr_Modi],2            ; für Return-Wert FN(ScrGet)
```

```
P_TEXT:  CALL    FlagClr
         CALL    Set_Mode
         MOV     ES,[ES_Stack]
         PUSHF                          ; - wegen RETI nachste Zeile
         CALL    DWord Ptr [Old_Int10]
         MOV     AH,00101001B
         MOV     AL,BYTE PTR [Scr_Mode_Set]
         AND     AL,00000100B
         OR      AL,AH                  ; siehe Kommentar bei P_GRAF
         MOV     [Scr_Mode_Set],AL
         CMP     [CGA_Mode_Set],mono
         JNE     CGA_TXT
         MOV     [CGA_Mode_Set],AL
CGA_TXT: MOV     DX,ctrl_reg            ; zurück zum Hercules-Textmodus
         OUT     DX,AL
         CALL    FlagClr                ; sicherheitshalber
         stc                            ; (Fehlerflag, wegen Modus 7)
         JMP     RETINT
;-----------------------------------------------
;- Punkt setzen je nach Flag                   -
;-----------------------------------------------
Pnt_Jmp  PROC    NEAR
         PUSH    ES
         PUSH    BP
         MOV     AX,Scrn1_Base
         MOV     ES,AX                  ; Bildschirmsegment
         MOV     BP,90
         CMP     [Hercules_Mod],true    ; IF   Herculespunkt setzen
         JE      HrcSPnt                ; THEN setze Hercules-Punkt
         MOV     BP,80
         CMP     [Oli_Mod],true         ; IF   Olivetti-Punkt setzen
         JE      EmulPnt                ; THEN setzen
         JMP     CGA_Pnt                ; ELSE setze CGA-Punkt
;-----------------------------------------------
;- Punkt setzen im Emulations-Modus 640x400    -
;-----------------------------------------------
EmulPnt  PROC    NEAR
         CALL    GetAdr
         OR      ES:[BX],AL    ; setze Bit
         POP     BP
         POP     ES
         JMP     RETINT
EmulPnt  ENDP
;-----------------------------------------------
;- Punkt setzen im Hercules-Grafik-Mode        -
;-----------------------------------------------
HrcSPnt  PROC    NEAR
         CALL    GetAdr
         OR      ES:[BX],AL    ; setze Bit
         POP     BP
         POP     ES
         JMP     RETINT
HrcSPnt  ENDP
;-----------------------------------------------
;- Punkt setzen im Emulations-Modus 640x200    -
;- d.h. jede dritte Zeile ist schwarz          -
;-----------------------------------------------
CGA_Pnt  PROC    NEAR
COMMENT $ Wir müssen die Adresse wegen 3. Zeile sowieso berechnen
         PUSHF
         CALL    DWord Ptr [Old_Int10] ; Punkt setzen über INT10
; $
         CALL    CGAAdr
         OR      ES:[BX],AL            ; setze Bit
CPLP:    ADD     BX,2000H
         CMP     BX,4000H              ; IF   noch nicht 3. Zeile
         JB      CPLP                  ; THEN nächste Zeile
         AND     AL,ES:[BX-4000H]      ; Hole Punkt von 1. Zeile
         AND     AL,ES:[BX-2000H]      ; Hole Punkt von 2. Zeile
         OR      ES:[BX],AL            ; setze ggf. in  3. Zeile
         POP     BP
         POP     ES
         JMP     RETINT
CGA_Pnt  ENDP
Pnt_Jmp  ENDP
;-----------------------------------------------
;- Punkt holen  je nach Flag                   -
;-----------------------------------------------
Pnt_Hol  PROC    NEAR
         MOV     AX,Scrn1_Base
         PUSH    ES
         PUSH    BP
         MOV     AX,Scrn1_Base
         MOV     ES,AX                 ; Bildschirmsegment
         MOV     BP,90
         CMP     [Hercules_Mod],true   ; IF   Herculespunkt setzen
         JE      HolePnt               ; THEN setze Hercules-Punkt
         MOV     BP,80
         CMP     [Oli_Mod],true        ; IF   Olivetti-Punkt setzen
         JE      HolePnt               ; THEN holen
         POP     BP
         POP     ES
         JMP     DWord Ptr [Old_Int10] ; ELSE hole CGA-Punkt
;-----------------------------------------------
;- Punkt holen für Hercules-/Olivetti-Grafik   -
;-----------------------------------------------
HolePnt  PROC    NEAR
         CALL    GetAdr
         AND     AL,ES:[BX]            ; hole Bit
         POP     BP
         POP     ES
         JMP     RETINT
HolePnt  ENDP
Pnt_Hol  ENDP
;-----------------------------------------------------
;- Einsprung mit 'Spalte in CX',  'Zeile in DX'
;-              'Faktor in BP',  'Bildschirmsegment in ES'
;-              ( etwas Mathe:     2000 Hex = 8192 dezimal = 2^13 )
;- A) Byte-Position für CGA-Emulation          (nur Hercules Seite 2)
;- B) Byte-Position für Olivetti-Emulation     (nur Hercules Seite 2)
;- C) Byte-Position für Hercules (beide Seiten: (Z.Zt. nur Seite 2))
;- a) Byteadresse:= 8192 *(Y mod 2) + 80*INTEGER(Y/2) + INTEGER(X/8)
;- b)             := 8192 *(Y mod 4) + 80*INTEGER(Y/4) + INTEGER(X/8)
;- c)             := 8192 *(Y mod 4) + 90*INTEGER(Y/4) + INTEGER(X/8)
;- Bit-Position errechnet sich in allen 3 Fällen:  128 SHR (X mod 8)
;-----------------------------------------------------
CGAAdr   PROC    NEAR
         MOV     BX,DX          ; (Y MOD 2)
         AND     DX,00000001B
         PUSH    CX
         MOV     CL,13          ; 8192 = Zeileninterlace
         SHL     DX,CL
         POP     CX
         PUSH    DX
         SHR     BX,1           ; 80 * INTEGER(Y/2)
         MOV     AX,BP
         IMUL    BX
         POP     DX
```

```
          ADD     DX,AX
          JMP     Bit_Adr       ; RETURN von dort
CGAAdr    ENDP
;
GetAdr    PROC    NEAR
          MOV     BX,DX         ; 2000H * (Y MOD 4)
          AND     DX,00000011B
          MOV     AX,2000H      ; Zeileninterlace
          IMUL    DX
          PUSH    AX
          SHR     BX,1          ; .. * INTEGER(Y/4)
          SHR     BX,1
          MOV     AX,BP         ; 90/80 == Hercules/Olivetti
          IMUL    BX
          POP     DX
          ADD     DX,AX
          JMP     Bit_Adr       ; RETURN von dort
GetAdr    ENDP
;
Bit_Adr   PROC    NEAR
          MOV     BX,CX         ; (noch Adressenberechnung)
          SHR     BX,1          ; INTEGER(X/8)
          SHR     BX,1
          SHR     BX,1
          ADD     BX,DX
;                               ; (ab hier Bitberechnung)
          AND     CL,0111B      ; Bitnummer: 128 SHR (X mod 8)
          MOV     AL,128
          SHR     AL,CL         ; Zuruck mit flgd. Werten:
          RET                   ; AL := BIT OF WORD OF ES:[BX]
Bit_Adr   ENDP
;-----------------------------------------
;
;-----------------------------------------
UTILITIE  PROC    NEAR
ClrScr:   PUSH    ES                    ; Grafikschirm loschen
          PUSH    DI
          PUSH    CX
          PUSH    AX
          MOV     CX,4000H              ; 16 kB
          XOR     AX,AX
          MOV     DI,SCRN1_BASE
          MOV     ES,DI
          XOR     DI,DI
          CLD
REPNZ     STOSW
          POP     AX
          POP     CX
          POP     DI
          POP     ES
          RET
;
Set_Crt:  PUSH    AX                    ; Schirminit (INT 1DH)
          PUSH    CX
          CALL    Sync
          MOV     SI,word PTR [param_ptr]
          MOV     CX,16D                ; # Parameter und Register
          XOR     Ax,Ax                 ; starte bei Register 0
          MOV     DX,indx_reg           ;        über Index-Register
          CLD
SI_Lp:    MOV     AL,AH
          OUT     DX,AL                 ; Registerwahl
          INC     DX                    ; vorbereiten für Wertübergabe
          LODSB                         ; AL:=[SI], SI:=SI+1
          OUT     DX,AL                 ; Wert in Register schreiben
          DEC     DX                    ; vorbereiten für Registerwahl
          INC     AH                    ; nachstes Register
LOOP      SI_Lp                         ; CX:=CX-1, Z:=(CX==0)
          POP     CX
          POP     AX
          RET
;
Sync:     PUSH    AX                    ; Vertical-Sync-Signal
          PUSH    BX
          PUSH    DX
          MOV     DX,status_reg
          XOR     BX,BX
Sync1_Lp: IN      AL,DX
          TEST    AL,10000000B          ; (Anzeige) Vertical Sync
          JNZ     Sync2_Lp
          DEC     BX
          JNZ     Sync1_Lp
Sync2_Lp: AND     BX,BX
          JZ      Sync_End
          XOR     BX,BX
Sync3_Lp: IN      AL,DX
          TEST    AL,10000000B
          JZ      Sync_End
          DEC     BX
          JNZ     Sync3_Lp
Sync_End: POP     DX
          POP     BX
          POP     AX
          RET
UTILITIE  ENDP
;-----------------------------------------
;- Schirm ausblenden, warten, ein        -
;-----------------------------------------
Screen_U  PROC    NEAR
          CALL    Scr_Off
          CALL    Lp_Lag
          JMP     Scr_On                ; Return von dort
;
Scr_Off:  PUSH    AX                    ; Schirm aus
          PUSH    DX
          MOV     AL,AH
          AND     AL,NOT Scr_On_Flag
          MOV     DX,ctrl_reg
          OUT     DX,AL
          POP     DX
          POP     AX
          RET
;
Lp_Lag:   PUSH    CX                    ; Verzögerungsschleife
          XOR     CX,CX                 ; == mov cx,0000
Do_Lplg:  DEC     CX
          JNZ     Do_Lplg
          POP     CX
          RET
;
Scr_On:   PUSH    AX
          PUSH    DX
          MOV     AL,AH                 ; Schirm ein
          OR      AL,Scr_On_Flag
          MOV     DX,ctrl_reg
          OUT     DX,AL
          MOV     [Scr_Mode_Set],AL
          CALL    SetCRTMod
```

```
        POP     DX
        POP     AX
        RET
Screen_U ENDP
;-----------------------------------------------
; BIOS-Daten aus Programm-Datenspeicher setzen
;-----------------------------------------------
ASSUME DS:Bios_Data
Set_Bios PROC   NEAR
        CALL    FlagSet
        CALL    Set_Mode
        JMP     Set_col                 ; Return von dort

SetCRTMod:PUSH  DS                      ; setze BIOS-Crt-Modusparameter
        PUSH    BX
        PUSH    AX
        MOV     AX,Bios_Data
        MOV     DS,AX
        MOV     AL,BYTE PTR CS:[Scr_Mode_Set]
        MOV     BX,OFFSET Bios_Data:CRT_Mode_Set
        MOV     [BX],AL
        JMP     BIOS_Ret
;
Set_Col: PUSH   DS                      ; setze BIOS_Spaltenbreite 40/80
        PUSH    BX
        PUSH    AX
        MOV     AX,Bios_Data
        MOV     DS,AX
        MOV     BX,OFFSET Bios_Data:CRT_Cols
        MOV     AL,BYTE PTR CS:[Scr_Cols]
        MOV     [BX],AL
        JMP     BIOS_Ret
;
Set_Mode:PUSH   DS                      ; setze BIOS-Crt-Modus
        PUSH    BX
        PUSH    AX
        PUSH    CX
        MOV     CL,AL
        MOV     AX,Bios_Data
        MOV     DS,AX
        MOV     BX,OFFSET Bios_Data:CRT_Mode
        MOV     [BX],CL
        POP     CX
        JMP     BIOS_Ret
;
FlagSet: PUSH   DS                      ; behaupte: CGA vorhanden
        PUSH    BX
        PUSH    AX
        MOV     AX,Bios_Data
        MOV     DS,AX
        MOV     BX,Offset Bios_Data:equIP_Flag
        AND     BYTE PTR [BX],11101111B
        JMP     BIOS_Ret
;
FlagClr: PUSH   DS                      ; normales FLAG
        PUSH    BX
        PUSH    AX
        MOV     AX,Bios_Data
        MOV     DS,AX
        MOV     BX,Offset Bios_Data:equIP_Flag
        OR      BYTE PTR [BX],00110000B
BIOS_Ret:POP    AX
        POP     BX
        POP     DS
        RET
Set_Bios ENDP
ASSUME DS:CSEG
NewInt10 ENDP
;-----------------------------------------------
;- INT11H liefert equipmentflag-Wort           -
;- Bits 5+4    Modus:   01 CGA 40x25 mono     -
;-                      10 CGA 80x25 mono     -
;-                      11 Monochrom-Karte    -
;-----------------------------------------------
equipm  PROC    NEAR
        PUSHF                           ; - wegen RETI nächste Zeile
        CALL    DWord Ptr CS:[Old_Int11]
        CMP     BYTE Ptr CS:[Scr_Mode_Set],mono
        JE      Eq_End                  ; S. Bemerkung 11
        AND     AX,1111111111011111B    ; S. Bemerkung 12
;        JE      Eq_End                 ; S. Bemerkung 11
        CALL    FlagSet                 ; S. Bemerkung 13
Eq_End: IRET
equipm  ENDP
;-----------------------------------------------
;- INT16H wird für die Kopierfunktion im       -
;-       Textmodus benötigt. Sie kopiert      -
;-       2000 Bytes = 1 Bildschirmseite von   -
;-       CGA-Textseite zur Mono-Textseite     -
;-       ( wenn vom Programm direkt dorthin   -
;-         geschrieben wird )                 -
;- liefert: AH:=Scan-Kode der Taste           -
;-          AL:=Zeichen                      -
;- (sicherer wäre Interrupt 9)                -
;-----------------------------------------------
CopyPage PROC   NEAR                    ; IF NOT im Mono-Modus
        CMP     BYTE Ptr CS:[Scr_Mode_Set],mono
        JNE     PC_End                  ; THEN   keine weitere Aufgabe
;        CMP     AH,02H                 ; ELSE   Test ob FN(0,1)
;        JB      PCopy
        OR      AH,AH                   ; ELSE   Test ob FN(0,1)
        JZ      PCopy
PC_End: JMP     DWord Ptr CS:[Old_Int16]
;
PCopy:  PUSHF                           ; - wegen RETI nächste Zeile
        CALL    DWord Ptr CS:[Old_Int16]
        CMP     AH,54H                  ; SysReq beim AT
;*      CMP     AL,                     ; hier nicht implementiert
;        JMP     EndCP
        JNE     EndCP
        CALL    Do_Copy                 ; Text von Segment B800 -> B000
        MOV     AX,0                    ; warte auf nächste Taste
        JMP     PCopy
;
EndCP:  IRET
;
Do_Copy PROC    NEAR
        PUSH    BP
        PUSH    ES
        PUSH    DS
        PUSH    BX
        PUSH    DI
        PUSH    SI
        MOV     AX,Scrn0_Base           ;  Zielsegment
        MOV     DS,AX
        MOV     AX,Scrn1_Base           ; Quellsegment
        MOV     ES,AX
        MOV     CX,2000                 ; Anzahl Zeichen mit Attribut
        XOR     AX,AX
```

```
        MOV     BX,AX
        MOV     SI,AX                   ; ES:SI
        MOV     DI,AX                   ; DS:DI
        CLD
COUNT:  MOV     BX,Word Ptr ES:[si]
        MOV     Byte Ptr [DI],BL
        INC     DI
        INC     DI
        MOV     Byte Ptr [DI],BH
        INC     DI
        INC     DI
        INC     SI
        INC     SI
LOOP    COUNT
        POP     SI
        POP     DI
        POP     BX
        POP     DS
        POP     ES
        POP     BP
        RET
Do_Copy ENDP
CopyPage ENDP
;-------------------------------------------------
SUBTTL  Installation
PAGE +
INSTALL PROC    NEAR
K_Rqst: MOV     AX,Kennung
        INT     Video
        CMP     AX,installed            ; IF   CGAHERC vorhanden
        JZ      K_End                   ; THEN AL:=false
        MOV     AL,True                 ; ELSE al:=true
K_End:  RET
;
Do_Init: PUSH   DS                      ; Initialisierung
        PUSH    ES
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     BX,CS
        MOV     ES,BX                   ; ES:=CS
        MOV     DS,BX                   ; DS:=CS
        CALL    K_Rqst                  ; Test ob bereits installiert
        OR      AL,AL                   ; IF   schon installiert
        JZ      NoVec                   ; THEN springe
        CALL    DoVec10                 ; ELSE Vektoren umbiegen
        CALL    DoVec11
;       CALL    DoVec16
        CALL    DoVec42
        MOV     AL,true                 ; Vermerke: Vektoren umgebogen
NoVec:  XOR     AL,true
        CLD
        MOV     BYTE PTR [KennFlag],AL; = NOT(K_Rqst)
        MOV     BX,DOS_Puffer
        MOV     CL,[BX]                 ; Lange des Parameters:   p'
        MOV     AL,[BX+02]              ; Char-Parameter
        AND     AL,11011111B            ; 'gztb' -> 'GZTB'
        DEC     CL
        DEC     CL
        JNZ     Err_Txt                 ; Parmeteranzahl/Parameter falsch
        CMP     AL,'G'                  ; 'G'RAFIK
        JNE     I_Z40
        CALL    CGA_Mode                ; schalte in Grafikmodus
        JMP     IniEnd
;
I_Z40:  CMP     AL,'Z'                  ; 40 'Z'eichen/Zeile
        JNE     I_Txt
        CALL    CGA_Z40                 ; schalte in Textmodus
        JMP     IniEnd
;
I_Txt:  CMP     AL,'T'                  ; 'T'EXT
        JNE     I_Boot
        CALL    Txt_Mode                ; schalte in Textmodus
        JMP     IniEnd
;
I_Boot: CMP     AL,'B'                  ; 'B'OOT
        JE      Boot
Err_Txt: MOV    DX,OFFSET Error1        ; Fehlertextausgabe
        MOV     AH,StrgOut
        INT     MsDos
IniEnd: POP     DX                      ; Ende von Do_Init
        POP     CX
        POP     BX
        POP     AX
        POP     ES
        POP     DS
        RET
;-----------------------------------------
;- Warmboot-Interrupt 19H                -
;-----------------------------------------
COMMENT $       Bei der Boot-Routine wird das Programm nicht resident ge-
        laden. Ist jedoch nicht schlimm, da die Interruptvektoren doch
        umgebogen werden. Deshalb nach einem Aufruf von 'CGAHERC G'
        oder 'CGAHERC T' nie 'CGAHERC B' aufrufen! Hier dann bitte das
        kleine Programm BOOT.COM benutzen. Bei einigen Programme wird
        evtl. sogar SOLOBOOT.COM (TM) benotigt. - Auf Grund der MS-DOS-
        (/BOOT-Programm-)Architektur kann es passieren, daß, CGAHERC im
        Speicher uberschrieben wird. Dies führt dann zum System-Absturz.
        Der Aufruf von  'SAVE.COM'  vor Aufruf aller anderen residenten
        Programme ( incl. KEYBGR.COM ! ) umgeht dieses Problem.
            Der Interrupt 19H (Boot) funktioniert nicht immer korrekt bei
        DOS-Versionen kleiner als V3.0
END $
Boot:   CALL    CGA_Mode
        MOV     DX,OFFSET COPY$         ; Textausgabe
        MOV     AH,StrgOut
        INT     MsDos
        MOV     DX,OFFSET MSG_Boot      ; Textausgabe
        MOV     AH,StrgOut
        INT     MsDos
        STI
Eing:   MOV     AH,GetChar              ; warte auf Tastatureingabe
        INT     Tast_Ein
        CMP     AL,Ctrl_C
        JNE     Spc_Test
        JMP     IniEnd
Spc_Test:CMP    AL,Space
        JNE     Eing
        MOV     DX,OFFSET MSG_Wait      ; Textausgabe
        MOV     AH,StrgOut
        INT     MsDos
        XOR     AX,AX                   ; Register sicherheits-
        MOV     BX,AX                   ;         halber loschen
        MOV     CX,AX                   ; (XOR 3, MOV 2 Taktzyklen)
        MOV     DX,AX
        INT     Boot_Int
;=======================================
;= Nicht-Residenter Datenbereich       =
;=======================================
MSG_Boot DB     0DH,0AH,'Legen Sie Ihre Boot-Diskette in Laufwerk A:
        DB      0DH,0AH,'Booten  = Dann Leertaste (SPACE) druecken!'
        DB      0DH,0AH,'ABBRUCH = CTRL-C',0DH,0AH
        DB      0DH,0AH,'Bei Problemen - alle residenten Programme entfernen
```

```
(KeyBGR)',
          DB        0DH,0AH,'                - SAVE.COM aufrufen!'
          DB        0DH,0AH,'                - Jetzt die notwendigen residenten
Programme laden!'
          DB        0DH,0AH,'                - Evt. mit (SOLO)BOOT.COM laden, vorher
CGAHERC aufrufen!'
          DB        0DH,0AH,'- Telefonische Information 0221-317232'
          DB        '$'
MSG_Wait  DB        0DH,0AH,0AH,0AH,'... Bitte warten ...$'
MsgNoIn   DB        0DH,0AH,0DH,0AH,'CGAHERC nicht (erneut) installiert.$'
Error1    DB        'Ungueltiger Parameter (G,T,B):  bitte starten Sie erneut!$'
KennFlag  DB        0
;-----------------------------------------------
;- Bildschirmmodi 'T'ext und 'G'raf 640x200  -
;-        ab V3.1 'Z' zusätzlich 320x200     -
;-----------------------------------------------
Txt_Mode  PROC      NEAR                    ; T: Textmodus monochrom
          MOV       AH,07H
          JMP       Erstwahl
Txt_Mode  ENDP
;
CGA_Mode  PROC      NEAR                    ; G: 640x200
          MOV       AH,06H
          JMP       Erstwahl
CGA_Mode  ENDP
;
CGA_Z40   PROC      NEAR                    ; Z: 320x200
          MOV       AH,04H
CGA_Z40   ENDP                            ; Durchlauf zu Erstwahl
;
Erstwahl  PROC      NEAR
          MOV       AL,03                   ; Hercules FULL-Modus
          MOV       DX,CRT_Config           ; (Einmal-Aufruf nach Booten genügt)
          OUT       DX,AL
          MOV       AL,AH
          MOV       AH,ScrSel
          INT       Video
          RET
Erstwahl  ENDP
;-----------------------------------------------
;- Interrupt-Vektoren umbiegen      (DS==CS) -
;-       INT 10H  Bildschirm-Interrupt       -
;-       INT 11H  Hardwareausstattung        -
;-       INT 16H  Tastatur                   -
;-       INT 42H  EGA-BIOS-Zeiger            -
;-----------------------------------------------
DoVec10   PROC      NEAR
          PUSH      DS
          PUSH      ES
          PUSH      DX
          MOV       AL,Video                ; hole alten Int.-Vektor
          MOV       AH,GetVec               ;       in ES:BX
          INT       MsDos
          MOV       SI,offset Old_Int10     ; speichere alte Vektoradresse
          MOV       DX,offset NewInt10      ; Setze neuen Int.-Vektor
          MOV       AL,Video
          JMP       Vec_Ret
DoVec10   ENDP
;
DoVec11   PROC      NEAR
          PUSH      DS
          PUSH      ES
          PUSH      DX
          MOV       AL,Hardware             ; hole alten Int.-Vektor
          MOV       AH,GetVec               ;       in ES:BX
          INT       MsDos
          MOV       SI,offset Old_Int11     ; speichere alte Vektoradresse
          MOV       DX,offset equipm        ; Setze neuen Int.-Vektor
          MOV       AL,Hardware
          JMP       Vec_Ret
DoVec11   ENDP
;
DoVec16   PROC      NEAR
          PUSH      DS
          PUSH      ES
          PUSH      DX
          MOV       AL,Tast_Ein             ; hole alten Int.-Vektor
          MOV       AH,GetVec               ;       in ES:BX
          INT       MsDos
          MOV       SI,offset Old_Int16     ; speichere alte Vektoradresse
          MOV       DX,offset CopyPage      ; Setze neuen Int.-Vektor
          MOV       AL,Tast_Ein
DoVec16   ENDP                            ; Durchlauf zu Vec_Ret
;
Vec_Ret   PROC      NEAR
          MOV       [SI],BX                 ; speichere alten Vektor
          MOV       [SI+02],ES
          MOV       AH,SetVec               ; setze Vektor aus DS:DX
          INT       MsDos
          POP       DX                      ; Return für DoVec##
          POP       ES
          POP       DS
          RET
Vec_Ret   ENDP
;
DoVec42   PROC      NEAR                    ; manche Programme fragen ab,
          PUSH      DS                      ; ob der Pointer auf 0:0 steht!
          PUSH      ES
          PUSH      DX
          MOV       AL,EGA_BIOS             ; hole alten Int.-Vektor
          MOV       AH,GetVec               ;       in ES:BX
          INT       MsDos
          MOV       SI,offset Old_Int42     ; speichere alte Vektoradresse
          MOV       [SI],BX                 ; speichere alten Vektor
          MOV       [SI+02],ES
          MOV       DX,0                    ; Setze neuen Int.-Vektor
          MOV       DS,DX
          MOV       AL,EGA_BIOS
          MOV       AH,SetVec               ; setze Vektor aus DS:DX
          INT       MsDos
          POP       DX                      ; Return für DoVec##
          POP       ES
          POP       DS
          RET
DoVec42   ENDP                            ; Durchlauf zu Vec_Ret
;-----------------------------------------------
;- Programmstart und -ende                   -
;-----------------------------------------------
Beginn    PROC      NEAR
          CALL      Do_Init                 ; DS:=CS
          CMP       BYTE PTR [KennFlag],false
          JE        Resident
          MOV       DX,OFFSET MsgNoIn
          MOV       AH,StrgOut
          INT       MsDos
          INT       Pend                    ; Programm beenden
;
Resident: MOV       DX,OFFSET INSTALL
          INT       Terminate               ; Programm beenden, aber resident
Beginn    ENDP
INSTALL   ENDP
;
CSEG      ENDS
          END       START
```

```
NSAVE.COM
A
MOV AX,31
MOV DX,5000
INT 21

RCX
7
W
Q
```

Bild 2. SAVE muß vor allen anderen residenten Programmen gestartet werden

```
NHGRAF.COM
A
MOV AX,00F0
INT 10
INT 20

RCX
7
W
Q
```

Bild 5. HGRAF schaltet die Hercules-Karte im Grafik-Modus auf die zweite Seite um

```
NPOINT.COM
A
MOV   CX,00C0
MOV   DX,CX
PUSH  CX
PUSH  CX
PUSH  DX
MOV   AH,0C
MOV   AL,01
INT   10
POP   DX
POP   CX
DEC   DX
JNS   106
CLD
POP   CX
MOV   DX,00C0
LOOP  0103
INT   20

RCX
1C
W
Q
```

Bild 8. Ein kleines Testprogramm für die drei Modi (HGRAF, CGA, OLIVETTI)

```
NEMULOFF.COM
A
MOV AX,00F1
INT 10
MOV AX,00FF
INT 10
INT 20

RCX
0C
W
Q
```

Bild 3. Eine Utility zum Ausschalten des Emulators

```
NOLI.COM
A
MOV AX,0040
INT 10
INT 20

RCX
7
W
Q
```

Bild 6. OLI wählt den CGA-Modus 640x400 (Olivetti)

```
NEMULON.TXT
A
MOV AX,00F1
INT 10
MOV AX,0006
INT 10
INT 20

RCX
0C
W
Q
```

Bild 4. EMULON ist nach dem Aufruf von EMULOFF zum erneuten Aktivieren des Emulators nötig

```
NMONO.COM
A
MOV AX,00F1
INT 10
INT 20

RCX
7
W
Q
```

Bild 7. Das Programm MONO entspricht der Befehlsfolge EMULON, CGAHERC T

```
NBOOT.COM
A
XOR AX,AX
XOR BX,BX
XOR CX,CX
XOR DX,DX
INT 19
INT 20

RCX
0C
W
Q
```

Bild 9. Wenn das Emulations-Programm schon resident ist und dennoch gebootet werden soll, benötigt man BOOT

erst den Interrupt 11H ab. Dann muß man in der Prozedur EQUIPM die Zeile AND AL,11101111B vor die Zeile CMP BYTE Ptr ... setzen. Bei manchen Programmen ist es notwendig, den bedingten Sprung „JE EQ_END" ganz aus der Emulation zu entfernen.

Einige Programme prüfen den EGA-BIOS-Vektor (INT 42H) und verweigern jede weitere Zusammenarbeit. Da er bei der Herculeskarte nicht benötigt wird, (Achtung: Kompatible) setzt ihn das Programm auf 0000:0000H!

Einfache Installation

Der Treiber wird mit seinem Namen CGAHERC und einer der Optionen G, T oder B aufgerufen. Dabei darf nur ein Leerzeichen zwischen dem Namen und der Option stehen. Zulässig sind folgende Aufrufe:

- CGAHERC G: Wählt die Farbgrafikemulation mit 80 Zeichen/Zeile
- CGAHERC Z: Wählt die Farbgrafikemulation mit 40 Zeichen/Zeile
- CGAHERC T: Zurück zum 80 × 25 Textmodus.
- CGAHERC B: Bei der Bootoption sind einige Dinge zu beachten, auf die wir weiter unten eingehen. Wie beim Aufruf CGAHERC G mit Warmboot der Spieldisk des CGA-Programms (Interrupt 19H). Die Meldung MSG_BOOT wird ausgegeben.
- CGAHERC: Bei Angabe anderer oder keines Parameters gibt CGAHERC eine Fehlermeldung aus. Bei ersten Aufruf wird es resident geladen. Dies muß man wegen der späteren 'B'OOT-Option beachten!

Emulation von Programmen, die selbst booten

Bei Programmen, wie z. B. dem Flugsimulator V1.0 von Microsoft, ist einiges mehr zu beachten. Die neugebooteten Programme nehmen keine Rücksicht auf eventuell vorhandene residente Programme und überschreiben mit großer Wahrscheinlichkeit die Speicherbereiche, in denen sich das Emulationsprogramm befindet. Da die Interrupt-Vektoren 10H und 11H auch auf diese Adressen zeigen, kann alles mögliche durch eine falsche Zuordnung geschehen. Mit einem kleinen Trick kann man das Überschreiben von Speicheradressen verhindern. Man erzeugt mit dem Debugger aus der Textdatei SAVE.TXT (*Bild 2*) eine kleine COM-Datei. Die Leerzeile vor RCX ist wichtig! Der Debugger wird wie folgt aufgerufen:

DEBUG < SAVE.TXT

Mit der neu geschaffenen Datei SAVE.COM kann man Speicherplatz reservieren, um das sechzehnfache des in Register DX angegebenen Wertes. Für Spiele ist der angegebene Wert ausreichend. Erst nach dem Aufruf von SAVE werden die anderen residenten Programme geladen (KEYBGR.COM) und CGAHERC B aufgerufen.

Weitere Hilfsprogramme

Die *Bilder 3...9* enthalten weitere Textdateien, die man mit dem Debuger – wie bei SAVE.COM – in COM-Dateien umwandeln kann.

Nach dem Starten der Utility EMULOFF bleibt der CGA-Emulator zwar im Speicher, jedoch wird die Emulation ausgeschaltet. Mit den Programmen EMULON oder MONO kann man sie wieder einschalten.

POINT zeichnet auf dem Bildschirm ein Dreieck (LOOP 103) oder ein Viereck (LOOP 105).

Es gibt nur wenige CGA-Programme, die unter der Emulation nicht laufen. Einige Programme geben im Text-Modus Text im Segment B800H aus, der auf dem Bildschirm nicht sichtbar ist. Hier müssen die Eingaben „blind" erfolgen. Im Listing finden sich weitere Hinweise, wie CGAHERC an spezielle Programme angepaßt werden kann. □

Norbert Juffa

Schnelle Algorithmen für Sin und Cos

Ohne numerischen Coprozessor dauert die Berechnung von Sinus- und Cosinus-Werten relativ lange. Ein effektiver Algorithmus, der schnelle Integerarithmetik-Befehle verwendet, legt auch dem langsamsten PC Siebenmeilenstiefel an. Das beschriebene Verfahren wird im 8087 verwendet.

Die Berechnung von Sinus und Cosinus ist in Integerarithmetik durchaus sinnvoll. Dies wurde bereits in einem früheren Beitrag gezeigt [1]. Ein weitgehend unbekannter, aber sehr effizienter Algorithmus, der Winkelfunktionen mit beliebiger Genauigkeit und zudem noch simultan liefert, ist das CORDIC-Verfahren [2, 3]. Beispielsweise verwendet auch der NDP 8087 das CORDIC-Verfahren zur Berechnung aller transzendenten Funktionen [4]. Der Begriff CORDIC kommt aus dem Englischen und bedeutet Coordinate Rotation Digital Computer, d.h. Koordinatendrehen mit dem Computer. Ausgangspunkt ist die Überlegung, daß jeder Winkel zwischen 0 und 90 Grad sich mehr oder weniger genau als Summe und Differenz eines geeigneten Satzes von n Basiswinkeln a_k ausdrücken läßt.

Wenn man nun schrittweise den Winkel u, dessen Sinus und Cosinus zu bestimmen sind, aus allen Basiswinkeln zusammensetzt, kann man gleichzeitig Sinus und Cosinus schrittweise miterrechnen. Die Vorgehensweise ist in *Bild 2* verdeutlicht. Wenn $a_0 \ldots a_{n-1}$ ein Satz von Basiswinkeln ist und w_k die k-te Teilsumme von u sind, in der a_0 bis a_{k-1} aufsummiert werden, dann gilt für:

$$(1)\quad y_k = \cos(w_k)$$
$$x_k = \sin(w_k)$$

```
                        ; Sinus und Cosinusberechnung mittels CORDIC  (c) '87 N. Juffa
                        ; Version 21.06.87
                        ;
                        ; Die Routine SinCos berechnet gleichzeitig Sinus und
                        ; Cosinus eines Winkels zwischen 0 und 360 Grad. Die
                        ; Winkel müssen gemäß round(65536/360*winkel) skaliert
                        ; sein. Sinus und Cosinus werden als vorzeichenbehaftete
                        ; skalierte Integerzahlen geliefert, wobei der Funktionswert
                        ; sich nach Division durch 32768 ergibt.
                        ;
                        ; Input:    CX    skalierter Winkelwert
                        ; Output:   AX    Sinus
                        ;          DX    Cosinus
                        ;
                        ; Zerstört: AX, BX, CX, DX, SI, DI, BP


0000            cseg    segment   para public 'code'

                assume  cs:cseg

0000            SinCos  PROC      NEAR

0000 EB 1E              JMP short Start           ; Tabelle umgehen

0002 2000       Table:  DW        2000h           ; Tabellenwerte errechnen sich
0004 12E4               DW        12e4h           ; wie folgt:
0006 09FB               DW        09fbh           ; round(arctan(0.5^0)*65536/360)
0008 0511               DW        0511h           ;   .
000A 028B               DW        028bh           ;   .
000C 0146               DW        0146h           ; round(arctan(0.5^14)*65536/360)
000E 00A3               DW        00a3h           ; wobei alle Winkel in Grad sind.
0010 0051               DW        0051h           ; 65536/360 ist der Skalierungs-
0012 0029               DW        0029h           ; für alle Winkel im Programm.
0014 0014               DW        0014h           ;
0016 000A               DW        000ah           ;
0018 0005               DW        0005h           ;
001A 0003               DW        0003h           ;
001C 0001               DW        0001h           ;
001E 0001               DW        0001h           ;

0020 8B F9      Start:  MOV       DI,CX           ; skalierter Winkel in CX
0022 81 E1 C000         AND       CX,0c000h       ; Quadrant in CH, Zähler in CL=0
0026 81 E7 3FFF         AND       DI,3fffh        ; Winkel reduziert auf 0..90
002A 33 C0              XOR       AX,AX           ; x    := 0
002C 8B D8              MOV       BX,AX           ; oldx := 0
002E BA 4DBA            MOV       DX,4dbah        ; y    := $4dba
0031 8B EA              MOV       BP,DX           ; oldy := y
0033 E8 0000            CALL      Hier            ; Tabellen-
0036 5E         Hier:   POP       SI              ;   anfang
0037 83 EE 34           SUB       SI,((Offset CS:Hier)-(Offset CS:Table)); berechnen
003A F7 DF              NEG       DI              ; winkel := -winkel
003C 0B FF      Label1: OR        DI,DI           ; If winkel < 0 then
003E 7C 09              JL        Label2          ;    goto 2
0040 2B C5              SUB       AX,BP           ; x    := x - oldy
0042 03 D3              ADD       DX,BX           ; y    := y + oldx;
0044 2E: 2B 3C          SUB       DI,CS:[SI]      ; winkel:= winkel-table [zähler]
0047 EB 07              JMP short Label3          ; goto 3;
0049 03 C5      Label2: ADD       AX,BP           ; x    := x + oldy;
004B 2B D3              SUB       DX,BX           ; y    := y - oldx;
004D 2E: 03 3C          ADD       DI,CS:[SI]      ; winkel:= winkel+table [zähler]
0050 FE C1      Label3: INC       CL              ; zähler := zähler + 1
0052 83 C6 02           ADD       SI,0002         ; nächstes Tabellenelement
0055 8B DB              MOV       BX,AX           ; oldx := x
0057 8B EA              MOV       BP,DX           ; oldy := y
0059 0B DB              OR        BX,BX           ; if oldx < 0 then
005B 7C 04              JL        Label4          ;    goto 4
005D D3 FB              SAR       BX,CL           ; oldx := oldx div (2^zähler)
005F EB 06              JMP short Label5          ; goto 5
0061 F7 DB      Label4: NEG       BX              ; oldx := -oldx
0063 D3 FB              SAR       BX,CL           ; oldx := oldx div (2^zähler)
0065 F7 DB              NEG       BX              ; oldx := -oldx;
0067 0B ED      Label5: OR        BP,BP           ; if oldy < 0 then
0069 7C 04              JL        Label6          ;    goto 6
006B D3 FD              SAR       BP,CL           ; oldy := oldy div (2^zähler)
006D EB 06              JMP short Label7          ; goto 7
006F F7 DD      Label6: NEG       BP              ; oldy := -oldy
0071 D3 FD              SAR       BP,CL           ; oldy := oldy div (2^zähler)
0073 F7 DD              NEG       BP              ; oldy := -oldy
0075 80 F9 0F   Label7: CMP       CL,0fh          ; if zähler < 15 then
0078 7C C2              JL        Label1          ; goto 1;
007A D1 C1              ROL       CX,1            ; Quadrant in
007C D1 C1              ROL       CX,1            ;  CX (0..3)
007E 81 E1 0003         AND       CX,0003h        ;
0082 74 05              JZ        Label9          ; durch zyklisches
0084 92         Label8: XCHG      AX,DX           ;  Vertauschen der
0085 F7 DA              NEG       DX              ;   Zahlwerte und Vorzeichen
0087 E2 FB              LOOP      Label8          ;     an Quadranten anpassen
0089 C3         Label9: RET

                SinCos  ENDP

008A            cseg    ends
                        END
```

Bild 1: Das CORDIC-Verfahren realisiert in einem Assembler-Programm

$$(2)\ y_{k+1} = \cos(w_{k+1}) = \cos(w_k + a_k)$$
$$x_{k+1} = \sin(w_{k+1}) = \sin(w_k + a_k)$$

Nach den Additionstheoremen für Sinus und Cosinus folgt daraus:

$$(3)\ y_{k+1} = y_k \cdot \cos(a_k) - x_k \cdot \sin(a_k)$$
$$x_{k+1} = x_k \cdot \cos(a_k) + y_k \cdot \sin(a_k)$$

Dies ist aber die gleiche Formel, die beim Drehen eines Koordinatensystems angewandt wird, womit sich auch der Namen des Algorithmus erklärt. Wenn man berücksichtigt, daß sowohl Subtraktionen als auch Additionen von a_k vorkommen und daß $\tan(\alpha) = \sin(\alpha)/\cos(\alpha)$ ist, dann gewinnt man aus (3) durch Ausklammern von cos (a_k) ein Iterationsschema:

$$(4)\ y_{k+1} = \cos(a_k) \cdot (y_k \mp x_k \cdot \tan(a_k)) = \cos(w_{k+1})$$
$$x_{k+1} = \cos(a_k) \cdot (x_k \pm y_k \cdot \tan(a_k)) = \sin(w_{k+1})$$

Nach n Schritten (also nach Berücksichtigung aller Basiswinkel) ergibt sich das gewünschte Resultat:

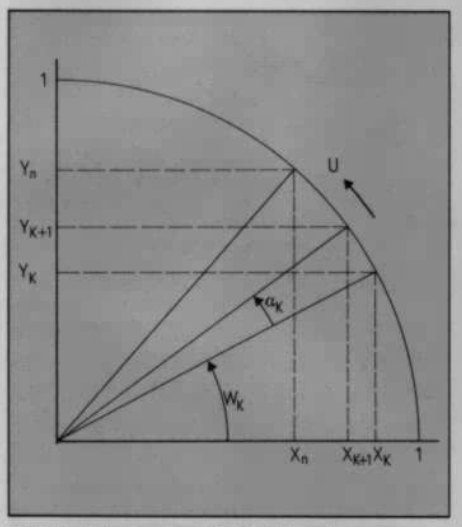


Bild 2: Prinzip des CORDIC-Verfahrens

$$(5)\ y_n = \cos(w_n) = \cos(u)$$
$$x_n = \sin(w_n) = \sin(u)$$

Allerdings ist dieses Iterationsschema noch viel zu kompliziert, um mit Integerarithmetik durchgeführt zu werden. Darum sei als erstes bei jedem Schritt die Multiplikation mit cos (a_k) weglassen:

$$(6)\ y_{k+1} = (y_k \mp x_k \cdot \tan(a_k))$$
$$x_{k+1} = (x_k \pm y_k \cdot \tan(a_k))$$

Daraus ergibt sich dann:

$$(7)\ y_n = 1/(\cos(\pm a_0) \cdot \ldots \cdot \cos(\pm a_n)) \cdot \cos(u)$$
$$x_n = \ldots \cdot \sin(u)$$

Diese häßliche Formel wird schöner, wenn man die Ausgangswerte x_0 und y_0 mit dem Faktor $f = \cos(\pm a_0) \cdot \ldots \cdot \cos(\pm a_n)$ multipliziert. Als Problem bleiben jetzt noch die Multiplikationen mit tan (a_k). Da man die Basiswinkel geeignet wählen kann, wählt man sie so, daß $\tan(a_k) = 0{,}5^{k-1}$. Damit reduzieren sich die Multiplikationen auf ein (k–1)-faches Rechtsschieben. Es sei $a_k = \arctan(0{,}5^{k-1})$ gesetzt. Diese Werte werden in einer Tabelle abgelegt. Für n = 15 ergibt sich f zu 0.60725. Nun liefert jeder Schritt eine gültige Binärstelle für das Endergebnis. Als Startwert für die Berechnung wähle man $x_0 = \sin(0) \cdot f = 0$ und $y_0 = \cos(0) \cdot f = 0.60725$.

Damit alles in Integerarithmetik ablaufen kann, müssen die Winkel und Funktionswerte noch skaliert werden. Für die Winkel wählt man zweckmäßigerweise den Faktor 65536/360, um den Vollkreis auf ein 16-Bit Wort abzubilden. Die Funktionswerte werden mit 2^{15} multipliziert, um alle Werte zwischen –1 und 1 darstellen zu können. Wenn man nun noch beachtet, daß negative Werte nicht gedankenlos durch Rechtsschieben geteilt werden dürfen (–13 div 4 = –3; –13 shr 2 = –4 !), so ergibt sich das Assemblerprogramm in *Bild 1*. Es ist in der Lage, auf einem mit 8 MHz getakteten 8086 Sinus und Cosinus in weniger als 0,5 ms zu errechnen.

Natürlich gibt es noch schnellere Verfahren, um Sinus und Cosinus zu berechnen. Ein solches, quadratische Interpolation in einer Tabelle, wurde bereits in mc angesprochen [5]. Dieses Verfahren leitet sich wie folgt her: Der Sinus wird in gleichmäßigen Winkelabständen tabuliert. Sucht man jetzt den Funktionswert eines Winkels, so bestimmt man, zwischen welchen Winkeln aus der Tabelle dieser Winkel liegt. Die Differenz zwischen dem kleineren dieser Werte und dem Winkel sei dx genannt. Durch die beiden Tabellenwerte und einen dritten legt man eine allgemeine quadratische Parabel ax^2+bx+c (*Bild 3*). Der Einfachheit halber werden der Nullpunkt auf den kleinsten der drei Tabellenwerte gelegt und dann nacheinander alle drei Werte in die Parabelgleichung eingesetzt. Man erhält folgendes Gleichungssystem:

$$(8)\ 0 \cdot a + 0 \cdot b + c = x$$
$$(9)\ 1 \cdot a + 1 \cdot b + c = y$$
$$(10)\ 4 \cdot a + 2 \cdot b + c = z$$

Aus (8) folgt sofort c = x, aus (9) folgt a = y – x – b. Einsetzen in (10) ergibt

$4 \cdot (y - x - b) + 2 \cdot b + x = z$, woraus folgt
$b = (4y - 3x - z) / 2$, also
$a = (x + z - 2y) / 2$.

In die so bestimmte quadratische Gleichung setzt man nun dx ein. Die Berechnung von a und b kann man durch Aufspalten einfacher gestalten. Eine auf diesem Verfahren aufbauende Routine (*Bild 4*) berechnet die Winkelfunktionen in ca. 0,16 msec, ist also dreimal so schnell wie die CORDIC-Routine. Allerdings wird bei diesem Verfahren für höhere Genauigkeit der Speicherbedarf für die Tabelle sehr groß, was wohl ein Grund war, für den 8087 das CORDIC-Verfahren zu wählen.

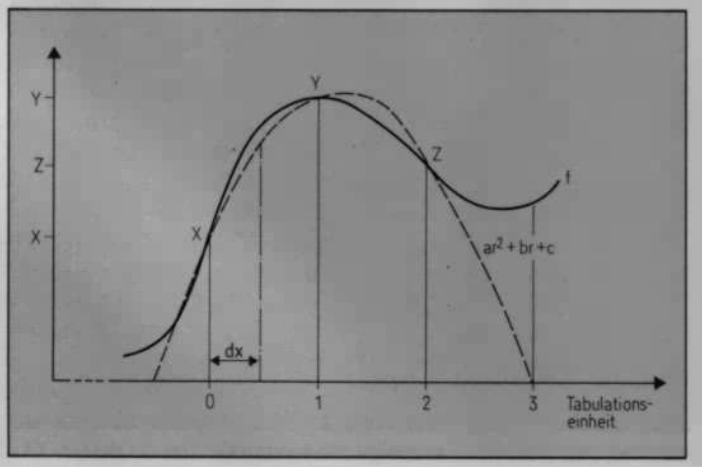


Bild 3: Interpolation von Sinus und Cosinus mit einer Parabel

Literatur

[1] *Obermaier, A.:* Sin und Cosin mit Integerarithmetik. mc 1987, Heft 6, Seite 108.

[2] *Völz, H.:* Algorithmen statt Tabellen. Urania 1984, Heft 11, Seite 44.

[3] *Hordeski, F. M.:* Microprocessor Cookbook. TAB Books, Blue Ridge Summit 1979

[4] *Palmer,Morse:* Die mathematischen Grundlagen der Numerikprozessoren 8087/80287. TEWI Verlag, München 1985

[5] *Phillip, J. E.:* Sin und Cosin. mc 1987, Heft 8, Seite 6

```
                        ; Sinus/Cosinus-Berechnung wie von J. Philipp in mc 8/87 vorgeschlagen
                        ;
                        ; (c) '87 N. Juffa                                    Version 16.1.88
                        ;
                        ; Die Routine SinCos berechnet den Sinus und Cosinus eines Winkels
                        ; zwischen 0 und 360 Grad in ca. 0,16 msec (8086/8Mhz). Die Winkel
                        ; müssen gemäß round (65536/360*winkel) skaliert sein. Sinus und
                        ; Cosinus werden als vorzeichenbehaftete skalierte Integerzahl ge-
                        ; liefert, wobei der Funktionswert sich nach Division durch 32768
                        ; ergibt.
                        ;
                        ; Input:    CX    skalierter Winkelwert
                        ; Output:   AX    skalierter Sinus
                        ;          CX    skalierter Cosinus
                        ;
                        ; Zerstört: AX, BX, CX, DX, SI, DI, BP


0000                    cseg    segment  para public 'code'

                        assume  cs:cseg

0000                    SinCos  PROC     NEAR

0000  EB 46                     JMP short Start          ; Tabelle umgehen

0002  0000              Table:  DW       00000h          ; Tabellenwerte errechnen sich
0004  0648                     DW       00648h          ; wie folgt:
0006  0C8C                     DW       00c8ch          ; round( 32768 * sin (0 * (180/64)))
0008  12C8                     DW       012c8h          ;   .
000A  18F9                     DW       018f9h          ;   .
000C  1F1A                     DW       01f1ah          ; round( 32768 * sin (34 * (180/64)))
000E  2528                     DW       02528h          ; wobei alle Winkel in Grad sind.
0010  2B1F                     DW       02b1fh          ;
0012  30FC                     DW       030fch          ;
0014  368A                     DW       036bah          ;
0016  3C56                     DW       03c56h          ;
0018  41CE                     DW       041ceh          ;
001A  471D                     DW       0471dh          ;
001C  4C40                     DW       04c40h          ;
001E  5133                     DW       05133h          ;
0020  55F5                     DW       055f5h          ;
0022  5A82                     DW       05a82h          ;
0024  5ED7                     DW       05ed7h          ;
0026  62F2                     DW       062f2h          ;
0028  66CF                     DW       066cfh          ;
002A  6A6D                     DW       06a6dh          ;
002C  6DCA                     DW       06dcah          ;
002E  70E2                     DW       070e2h          ;
0030  73B5                     DW       073b5h          ;
0032  7641                     DW       07641h          ;
0034  7884                     DW       07884h          ;
0036  7A7D                     DW       07a7dh          ;
0038  7C29                     DW       07c29h          ;
003A  7D8A                     DW       07d8ah          ;
003C  7E9D                     DW       07e9dh          ;
003E  7F62                     DW       07f62h          ;
0040  7FD8                     DW       07fd8h          ;
0042  7FFF                     DW       07fffh          ;
0044  7FD8                     DW       07fd8h          ;
0046  7F62                     DW       07f62h          ;
0048  E8 004B R         Start: CALL     Hier            ; Tabellen-
0048  5E                Hier:  POP      SI              ;   anfang
004C  83 EE 49                  SUB      SI,((Offset CS:Hier)-(Offset CS:Table)); berechnen
004F  8B EE                     MOV      BP,SI           ; Tabellenanfang retten
0051  E8 0065 R                 CALL     Routine         ; Sinus berechnen
0054  8B F5                     MOV      SI,BP           ; Tabellenanfang wiederholen
0056  8B E8                     MOV      BP,AX           ; Sinus retten
0058  8B CF                     MOV      CX,DI           ; Winkel wiederholen
005A  80 C5 40                  ADD      CH,40h          ; cos (x) = sin (x+90)
005D  E8 0065 R                 CALL     Routine         ; Cosinus berechnen
0060  8B C8                     MOV      CX,AX           ; CX = cosinus
0062  8B C5                     MOV      AX,BP           ; AX = sinus
0064  C3                        RET


0065  8B F9              Routine: MOV      DI,CX           ; Winkel retten
0067  0B C9                     OR       CX,CX           ; Winkel zwischen 180 und 360 ?
0069  79 02                     JNS      Noneg           ; nein
006B  F7 D9                     NEG      CX              ; ja, Winkel negieren
006D  80 FD 40           Noneg: CMP      CH,40h          ; Winkel im 2. Quadranten ?
0070  72 05                     JB       Anfang          ; nein
0072  F7 D9                     NEG      CX              ; ja, auf
0074  80 C5 80                  ADD      CH,80h          ;  1. Quadranten transformieren

0077  8A C5              Anfang: MOV      AL,CH           ; i = winkel shr 9
0079  25 00FE                   AND      AX,00feh        ;
007C  03 F0                     ADD      SI,AX           ;
007E  80 E5 01                  AND      CH,01h          ; dx = winkel and $1ff
0081  2E: 8B 44 04              MOV      AX,CS:[SI+04]   ; stab [i+2]
0085  2E: 8B 5C 02              MOV      BX,CS:[SI+02]   ; stab [i+1]
0089  2E: 8B 34                 MOV      SI,CS:[SI]      ; stab [i]
008C  2B C6                     SUB      AX,SI           ; stab [i+2] - stab [i]
008E  2B DE                     SUB      BX,SI           ; stab [i+1] - stab [i] = f1
0090  D1 E8                     SHR      AX,1            ; (stab [i+2] - stab [i])/2 = f2
0092  2B C3                     SUB      AX,BX           ; a = f2 - f1
0094  2B D8                     SUB      BX,AX           ; b = f1 - a
0096  F7 E9                     IMUL     CX              ; a*dx
0098  8A C4                     MOV      AL,AH           ; DX-AX 9-mal
009A  8A E2                     MOV      AH,DL           ;  rechts schieben
009C  D0 EE                     SHR      DH,1            ;    um Skalierung
009E  D1 D8                     RCR      AX,1            ;     auszugleichen
00A0  03 C3                     ADD      AX,BX           ; a*dx+b
00A2  F7 E9                     IMUL     CX              ; (a*dx+b)*dx
00A4  8A C4                     MOV      AL,AH           ; DX-AX 9-mal
00A6  8A E2                     MOV      AH,DL           ;  rechts schieben
00A8  D0 EE                     SHR      DH,1            ;    um Skalierung
00AA  D1 D8                     RCR      AX,1            ;     auszugleichen
00AC  03 C6                     ADD      AX,SI           ; stab [i] + (a*dx+b)*dx

00AE  0B FF                     OR       DI,DI           ; Winkel zwischen 180 und 360 grad ?
00B0  79 02                     JNS      Ende            ; nein, Sinus positiv
00B2  F7 D8                     NEG      AX              ; Sinus negieren
00B4  C3                 Ende:  RET                      ; Sinus fertig

                         SinCos ENDP

00B5                     cseg   ends
                                END
```

Bild 4: Schnelle Interpolation von Sinus- und Cosinus-Werten

Christian Neubauer

Einführung in Intuition

Teil 2: Gadgets

Dieser Teil unserer Serie beschäftigt sich mit den Gadgets, die das Intuition Reference Manual als „Arbeitspferde von Intuition" bezeichnet: mit diesem mächtigen Werkzeug kann man nämlich fast alles so programmieren, daß der Tastaturaufwand auf ein Minimum reduziert wird.

Gadgets nennt man Stellen auf dem Bildschirm, die eine Funktion auslösen, wenn man sie mit der Maus anklickt. Mit Gadgets kann man z. B. Screens verschieben, Bildschirmfenster in ihrer Lage und Größe verändern oder Funktionen von Anwenderprogrammen starten. Als Programmierer muß man dabei zwischen Anwendergadgets (engl. application gadgets) und den von Intuition zur Verfügung gestellten Systemgadgets unterscheiden. Die Installation von Systemgadgets wurde bereits im ersten Teil dieses Beitrages vorgestellt, diesmal soll das Augenmerk vor allem auf die Anwendergadgets gerichtet sein.

Anwendergadgets

Anwendergadgets lassen sich in vier Gruppen unterteilen:

- boolsche Gadgets
- proportionale Gadgets
- Gadgets für Zeichenketten (strings)
- Gadgets für ganze Zahlen (integer)

Boolsche Gadgets sind solche, die beim Anklicken die Meldung „Gadget wurde angeklickt" abschicken. Proportionale Gadgets sind die wohl jedem Amigabenutzer bekannten „Schieberegler", die z. B. Sonix (vertikal proportional) oder das Programm Preferences (horizontal und vertikal proportional) benutzt. Mit Zeichenketten-Gadgets kann man vom Benutzer Texteingaben holen, sie werden häufig bei Filerequestern – also der Frage nach einer oder mehreren Dateien – benutzt. Eng mit der letzten Gruppe verwandt sind die Zahlengadgets, nur daß hier die Eingabe einer Zahl statt eines Textes verlangt wird.

Im Gegensatz zu den Systemgadgets, deren Lage im Fenster festgelegt ist, können Anwendergadgets an jeder Stelle innerhalb des Fensters erscheinen. Auch ist ihre Form und Größe fast beliebig, und sie kann sich verändern, wenn man sie anklickt. Diese Eigenschaft nennt man im englischen Highlighting, wir werden darauf noch näher eingehen.

Die Gadget-Struktur

Wie schon die Screens und Fenster werden auch Gadgets durch eine Struktur *(Bild 1)* initialisiert. Im folgenden wird auf die Bedeutung der einzelnen Elemente dieser Struktur näher eingegangen.

NextGadget
NextGadget ist ein Zeiger auf das nächste Gadget in einer verketteten Liste von Gadget-Strukturen. In der letzten Struktur dieser Liste muß NextGadget auf Null gesetzt werden.

LeftEdge, TopEdge, Width, Height
Jedes Gadget hat nur einen bestimmten Bereich, in dem es „empfindlich" ist, d. h. das Gadget erkennt nur innerhalb dieses Bereiches, daß es angeklickt wurde. Der Bereich ist immer rechteckig. Mit den Werten LeftEdge und TopEdge wird immer die obere linke Ecke des Gadgets festgelegt; Width und Hight geben die Breite und Höhe an.

Flags
Mit den Flags wird das Verhalten eines Gadgets festgelegt. Dabei stehen folgende Flags zur Verfügung:

- ☐ GADGHCOMP
- ☐ GADGHBOX
- ☐ GADGHIMAGE
- ☐ GADGHNONE

Mit diesen vier Flags teilt man Intuition mit, wie das Gadget auf einen Maus-Klick reagieren soll. Setzt man GADGHCOMP, so

```
struct Gadget
{
    struct Gadget *NextGadget;  /* Naechstes Gadget in der Liste */

    SHORT LeftEdge, TopEdge;    /* "hit box" des Gadgets */
    SHORT Width, Height;        /* "hit box" des Gadgets */

    USHORT Flags;               /* Erklaerung siehe Text */

    USHORT Activation;          /* Erklaerung siehe Text */

    USHORT GadgetType;          /* Erklaerung siehe Text */

    /* Das Applicationprogramm kann hier angeben, wie das Gadget aus- */
    /* sehen soll. Die Variable GadgetRender sollte auf eine Image,   */
    /* eine Borderstruktur oder auf NULL zeigen.                      */
    APTR GadgetRender;


    /* Pointer auf eine Image-, oder Borderstruktur, die fuer das     */
    /* Highlighting verwendet wird.                                   */
    APTR SelectRender;

    struct IntuiText *GadgetText;   /* Text fuer dieses Gadget        */

    LONG MutualExclude; /* Diese Funktion wird erst seit kurzem unter-*/
                        /* stuetzt. Sie wurde deshalb nicht besprochen*/

    /* Dies ist der Pointer auf die Spezialstrukturen SpecialInfo und */
    /* PropInfo. Wird er nicht gebraucht, ->NULL setzen.              */
    APTR SpecialInfo;

    USHORT GadgetID;    /* Benutzervariable (von Intuition ignoriert) */
    APTR UserData;      /* Benutzervariable (von Intuition ignoriert) */
};
```

Bild 1. Mit einer solchen Struktur wird ein Gadget eröffnet

werden beim Anklicken alle Farbbits des Gadgets invertiert, das Gadget erscheint also in anderen Farben, verändert aber dabei seine Form nicht. Gibt man GADGHBOX ein, so zeichnet Intuition einen Rahmen um die Select Box. Bei GADGHIMAGE muß zusätzlich das Element SelectRender auf einen definierten Wert gesetzt werden: Setzten dieses Flags hat nämlich zur Folge, daß das Gadget seine Form beim Anklicken völlig verändert, statt des ursprünglichen Gadgets wird nun dasjenige angezeigt, das in SelectRender definiert wurde. Soll sich das Gadget hingegen weder in der Form noch in der Farbe ändern, so wird einfach GADGHNONE als Highlighting-Methode angegeben.
Neben diesen vier Flags existieren noch weitere sieben, von denen hier jedoch nur vier interessant sind:

☐ GADGIMAGE ☐ GRELRIGHT
☐ GRELBOTTOM ☐ SELECTED

Das Aussehen und die Farbgebung von Gadgets werden entweder durch sogenannte Bild- (= Image-) oder durch Rahmen- (= Border-)Strukturen bestimmt. Wie diese aufgebaut sind und wie man sie im Zusammenhang mit Gadgets benutzt, wird bei der Besprechung der Elemente GadgetRender und SelectRender kurz dargestellt. Um Intuition mitzuteilen, daß man ein Bild darstellen will, muß das Flag GADGIMAGE gesetzt werden. Anderenfalls nimmt Intuition an, daß GadgetRender und SelectRender auf Rahmenstrukturen zeigen.
Die nächsten beiden Flags sind für die Lage im Bildschirmfenster zuständig, insbesondere dafür, wie sich das Gadget bei Größenveränderungen dieses Fensters verhält. Normalerweise bezieht sich die Koordinate TopEdge auf die Fenster-Oberkante. Soll sie sich aber auf die untere Fensterbegrenzung beziehen, so muß man GRELBOTTOM angeben (Gadget RELative to BOTTOM). Analog dazu bezieht sich LeftEdge normalerweise auf die linke Seite des Fensters, nach Eingabe von GRELRIGHT jedoch auf die rechte Seite.
Mit dem letzten hier besprochenen Flag SELECTED kann man den Schaltzustand eines Gadgets einstellen: Setzt man das Flag, so ist das Gadget „eingeschaltet".

ActivationFlags
Auch diese Flags sind für das Verhalten des Gadgets gegenüber Maus-Klicks zuständig:

☐ TOGGLESELECT

Es gibt zwei unterschiedliche boolsche Gadgets: Die einen lösen nur eine Funktion aus (Tastschalter), die anderen geben einen Schaltzustand an (Kippschalter). Setzt man TOGGLESELECT, so erhält man ein Gadget, daß sich wie ein Kippschalter verhält, im anderen Fall eines von der Art eines Tastschalters.

☐ GADGIMMEDIATE
☐ RELVERIFY

Diese beiden Activation-Flags steuern den Zeitpunkt, zu dem ein Programm davon unterrichtet wird, daß der Benutzer überhaupt ein Gadget angeklickt hat. Setzt man GADGIMMEDIATE, so erhält man sofort eine Meldung, bei RELVERIFY (RELease VERIFY) sendet Intuition die Meldung erst ab, wenn der Benutzer die linke Maustaste auch wieder über dem Gadget losgelassen hat. Damit hat ein schneller Benutzer auch die Möglichkeit, sich noch anders zu entscheiden, nachdem er das Gadget angeklickt, aber noch nicht losgelassen hat: Er braucht nur vom Gadget „wegzufahren" und erst dann loszulassen, dann erhält das Programm keine Meldung.

☐ FOLLOWMOUSE

Nach Setzen dieses Flags erhält man auch dann Maus-Meldungen, wenn das Gadget bereits angeklickt ist.

Die drei folgenden Activation-Flags beziehen sich auf Gadgets für Zeichenketten und Zahlen:

☐ STRINCENTER
☐ STRINGRIGHT
☐ LONGINT

Mit den ersten beiden Flags wird die Positionierung einer Zeichkette im Gadget vorgegeben. Es stehen folgende Möglichkeiten zur Verfügung: Linksbündig (dazu wird gar nichts angegeben), zentriert (STRINGCENTER) und rechtsbündig (STRINGRIGHT). Mit LONGINT weist Intuition die Eingabe aus einem Zeichenketten-Gadget einer (vorzeichenbehafteten) 32-Bit-Integervariablen zu.

```
/* Die IntuiTextstruktur wird haeufig in einer verketteten Liste verwendet*/
/* Die Koordinaten an denen der String erscheint beziehen sich immer auf  */
/* die obere linke Ecke des beinhaltenden Elements. Der String muss mit 0 */
/* abgeschlossen werden.                                                  */

struct IntuiText
{
    UBYTE FrontPen, BackPen;     /* Farbreg.Nummern fuer Schrift/Hintergr. */
    UBYTE DrawMode;              /* hier steht der DrawMode                */
    SHORT LeftEdge;              /* relative X-Koordinate                  */
    SHORT TopEdge;               /* relative Y-Koordinate                  */
    struct TextAttr *ITextFont; /* NULL, fuer den Standardzeichensatz      */
    UBYTE *IText;                /* Pointer auf den Text                   */
    struct IntuiText *NextText; /* Pointer auf die naechste Struktur       */
};
```

Bild 2. Die in Bild 1 erwähnte Intui-Text-Struktur

GadgetType
Wie schon gesagt stehen vier Gadget-Typen zur Verfügung. Zur Unterscheidung muß man bei GadgetType einen der folgenden Typen angeben:

☐ BOOLGADGET
☐ STRGADGET
☐ PROPGADGET

Wenn Sie sich jetzt fragen, warum für vier Gadget-Typen nur drei Einstellungen zur Verfügung stehen, dann sollten Sie sich den letzten Abschnitt nochmal ansehen: Das Activation-Flag LONGINT macht aus einem Gadget für Zeichenketten eines für Zahlen.

GadgetRender
An dieser Stelle muß ein Zeiger auf eine Bild- (= Image-) bzw. Rahmen- (= Border-) Struktur angegeben werden, je nachdem, ob man das Flag GADGIMAGE gesetzt oder gelöscht hat. Will man weder das eine noch das andere anzeigen, so kann dieser Zeiger auf NULL gesetzt werden.

SelectRender
Dieser Zeiger muß auf den gleichen Datentyp wie GadgetRender zeigen – also auf eine Bildstruktur, wenn GadgetRender auf eine Bildstuktur zeigt, sonst auf eine Rahmenstruktur. SelectRender zu setzen hat überhaupt nur im Zusammenhang mit dem Flag GADGHIMAGE einen Sinn, da nur dann das „alternative" Gadget angezeigt wird.

GadgetText
Soll unter dem Gadget ein Text erscheinen, so kann man diesen Zeiger auf eine Intuition-Textstruktur *(Bild 2)* setzen. Die Koordinaten dieser Struktur beziehen sich auf die linke obere Ecke des Gadgets.

MutualExclude
Diese Funktion wird zur Zeit noch nicht unterstützt.

SpezialInfo
Alle Gadgets außer dem boolschen benötigen noch einige zusätzliche Angaben, die – wie immer beim Amiga – in einer Struktur festzulegen sind. SpezialInfo ist der Zeiger auf diese Struktur.

GadgetID
Dieses Feld ist nur für den Programmierer von Interesse: Hat ein Programm mehrere Gadgets im gleichen Fenster, so kann man hier eine individuelle Nummer für jedes dieser Gadgets ablegen und so sehr leicht abfragen, von welchem Gadget die Meldung kam. Intuition selbst ist der Inhalt dieses Elements völlig gleichgültig.

UserData
Auch dieses Element wird von Intuition ignoriert und ist nur für den Einsatz durch den Programmierer gedacht.

Die Spezialstrukturen

Das war jetzt wieder etwas viel auf einmal. Trotzdem ist es noch nicht genug: man muß sich zusätzlich noch mit den Spezial-Strukturen auseinandersetzen, da ohne diese nur boolsche Gadgets erzeugt werden können.
Als erste ist in *Bild 3* die StringInfo-Struktur abgebildet. Hier müssen nur die ersten fünf Elemente initialisiert werden:

Buffer
Dies ist ein Zeiger auf eine mit dem NULL-Byte abgeschlossene Zeichekette, die in dem Gadget steht. Ändert sie der Benutzer, so wird die alte Zeichenkette in den Undo-Puffer kopiert und die geänderte in den aktuellen Puffer geschrieben.

UndoBuffer
Hier steht ein Zeiger auf die Zeichenkette, die als vorletzte in dem Gadget stand.

BufferPos
Der hier angegebene Wert kennzeichnet die Stelle des Puffers, an der der Cursor anfangs steht.

MaxChars
Dieses Element enthält den Maximalwert der Zeichen, die in den Puffer passen, einschließlich der abschließenden Null.

DispPos
Mit dem Wert dieses Elementes wird das Zeichen bestimmt, das als erstes in Gadget angezeigt wird.

```
struct StringInfo
{
    /* Diese Variablen muessen initialisiert werden, alles Weitere er-   */
    /* digt Intuition.                                                    */
    UBYTE *Buffer;       /* Pointer auf den Puffer, fuer den String       */
    UBYTE *UndoBuffer;   /* Optionaler Puffer (UnDobuffer)                */
    SHORT BufferPos;     /* Zeichenposition im Puffer                     */
    SHORT MaxChars;      /* Maximaler Zeicheninhalt des Puffers (incl. 0) */
    SHORT DispPos;       /* Pufferposition des ersten angez. Zeichens     */

    /* Die folgenden Variablen managed Intuition.                         */
    SHORT UndoPos;       /* Zeichenposition im UnDo-Puffer                */
    SHORT NumChars;      /* Zahl der momentan im Puffer befindl. Zeichen  */
    SHORT DispCount;     /* Zahl der momentan sichtbaren Zeichen          */
    SHORT CLeft, CTop;   /* Offset zur oberen linken Ecke des Containers  */
    struct Layer *LayerPtr;      /* RastPort, der das Gadget beinhaltet.  */

    /* In diesem Feld steht der Zahlenwert, den der Benutzer eingegeben   */
    /* hat, falls das Gadget ein IntegerGadget ist. Man kann hier auch    */
    /* einen Startwert festlegen. (Damit kann man feststellen, ob der Be-*/
    /* nutzer ueberhaupt einen neuen Wert eingegeben hat.)                */
    LONG LongInt;

    /* Wenn man eine andere Tastaturbelegung fuer sein Gadget wuenscht,   */
    /* so setzt man ALTKEYMAP bei den ActivationFlags und einen Pointer   */
    /* auf seine KeyMap in dieses Feld.                                   */
    struct KeyMap *AltKeyMap;
};
```

Bild 3. Für Text- und Zahlen-Gadgets braucht man zusätzlich diese Spezialstruktur

```
struct PropInfo
{
    USHORT Flags;        /* Erklaerung siehe Text */

    /* Diese Variablen werden initialisiert, bevor das Gadget aktiviert  */
    /* wird.Spaeter kann man hier nach den Werten sehen, die der Benutzer*/
    /* eingestellt hat.(Auch wenn dieser gerade mit dem Gadget "spielt".)*/
    /* Um die Werte zu aendern, nachdem das Gadget aktiviert wurde, muss */
    /* ModifyProp() benutzt werden.                                      */
    /* Steht in HorizPot/VertPot 0, so bedeuted dies, dass das Gadget am */
    /* unteren Anschlag steht, $FFFF bedeudet 100%, das Gadget steht also*/
    /* am oberen Anschlag.                                               */
    USHORT HorizPot;     /* 16-bit horizontaler Wert (0-$FFFF)           */
    USHORT VertPot;      /* 16-bit verticaler Wert (0-$FFFF)             */

    /* Diese 16-bit Variablen legen fest, welcher Prozentsatz der Gesamt-*/
    /* information durch das Gadget tatsaechlich dargestellt wird.       */
    /* Die Variablen werden im Zusammenhang mit der AUTOKNOB Routine ver-*/
    /* wendet um die Groesse des Knopfes entsprechend der angezeigten    */
    /* Information einzustellen. Damit wird auch festgelegt wie weit sich*/
    /* der Knopf bewegt, wenn man in die GadgetselectBox klickt.         */
    /* Beispiel: (s.Text)                                                */
    /* Um die Variablen nachdem das Gadget angezeigt wird zu aendern,    */
    /* kann man ModifyProp() benutzen.                                   */

    USHORT HorizBody;            /* horizontal Body   */
    USHORT VertBody;             /* vertical Body     */

    /* Fuer diese Variablen sorgt Intuition, man sollte sie 0 setzen */
    USHORT CWidth;               /* Container Breite  */
    USHORT CHeight;              /* Container Hoehe   */
    USHORT HPotRes, VPotRes;     /* PotiIncrements    */
    USHORT LeftBorder;           /* Container Rand    */
    USHORT TopBorder;            /* Container Rand    */
};
```

Bild 4. Diese Spezial-Struktur ist für proportionale Gadgets nötig: Hier wird z. B. angegeben, ob es sich um einen horizontalen oder vertikalen Schieberegler handelt

Das war's schon für die Zeichenketten-Gadgets. Die Spezialstruktur proportionaler Gadgets zeigt *Bild 4*. Die Elemente haben folgende Bedeutung:

Flags
An zusätzlichen Flags in dieser Struktur stehen zur Verfügung:

- ☐ AUTOKNOB
- ☐ FREEHORIZ
- ☐ FREEVERT

Möchte man, daß sich Intuition um das Aussehen des Knopfes kümmert, so muß man das erste dieser Flags angeben. Die beiden anderen legen fest, ob er horizontal oder vertikal bewegt werden soll.

HorizPot, VertPot
In diesen beiden Elementen steht der Wert, auf dem das propotionale Gadget gerade steht. Die beiden Variablen sind vom Typ USHORT, daher liegt ihr Inhalt im Wertebereich $0...$FFFF. Steht das Gadget am

```
#include <intuition/intuition.h>
#include <dos.h>


/* Vorwaertsdeklarationen der Funktionen */

void clicked(USHORT);
void readdir(char*);
void printdir(int);
char FileBuf[40],UndoBuf[40],DrwBuf[40]="\x00";
int  FileCount, Offset;

/* Auf die Kommentierung jedes Elementes einzeln wurde ver-  */
/* zichtet, da ihre Bedeutung in den Bildern nachzulesen ist */

SHORT brd0dat[] = { 0,0,182,0 };
SHORT brd1dat[] = { -1,-1,54,-1,54,10,-1,10,-1,-1 };

/* Die naechsten vier Structs muessen(!) global stehen */

struct IntuitionBase *IntuitionBase;
struct GfxBase *GfxBase;
struct Window *wdw1;
struct FILEINFO info[50],oldinfo;

struct IntuiText IText = { 2,0,JAM2,0,0,0,NULL,0 };
struct IntuiText Text7 = { 3,0,JAM1,5,1,0,"OK\x00",0 };
struct IntuiText Text6 = { 3,0,JAM1,5,1,0,"DF0:\x00",0 };
struct IntuiText Text5 = { 3,0,JAM1,5,1,0,"DF1:\x00",0 };
struct IntuiText Text4 = { 3,0,JAM1,3,1,0,"Cancel\x00",0 };
struct IntuiText Text3 = { 2,0,JAM1,-66,0,0,"File:\x00",0 };
struct IntuiText Text2 = { 0,0,JAM1,0,0,0,"\x00",0 };
struct IntuiText Text1 = { 2,0,JAM1,-66,0,0,"Drawer:\x00",0 };



struct Border brd0 =
{ 0,11,3,0,JAM1,2,brd0dat,NULL };

struct Border brd1 =
{ 0,0,2,0,JAM1,5,brd1dat,NULL };

struct Border brd2;


struct Gadget Gadg[11];    /* Platz fuer die 10 Boolgadgtes */
                           /* "unter" den Filenamen         */



/* Hier werden die vier Bool, zwei String und das      */
/* PropGadget initialisiert                            */

struct Gadget gdg7 =
{ NULL,18,157,54,10,GADGHCOMP,RELVERIFY,BOOLGADGET,
  &brd1,NULL,&Text7,NULL,NULL,7,NULL };

struct Gadget gdg6 =
{ &gdg7,90,157,54,10,GADGHCOMP,RELVERIFY,BOOLGADGET,
  &brd1,NULL,&Text6,NULL,NULL,6,NULL };

struct Gadget gdg5 =
{ &gdg6,162,157,54,10,GADGHCOMP,RELVERIFY,BOOLGADGET,
  &brd1,NULL,&Text5,NULL,NULL,5,NULL };

struct Gadget gdg4 =
{ &gdg5,234,157,54,10,GADGHCOMP,RELVERIFY,BOOLGADGET,
  &brd1,NULL,&Text4,NULL,NULL,4,NULL };

struct StringInfo spez3 =
{ &FileBuf[0],&UndoBuf[0],0,30,0,0,0,0,0,0,0,0,0 };

struct Gadget gdg3 =
{ &gdg4,85,130,190,10,NULL,RELVERIFY|TOGGLESELECT,STRGADGET,
  &brd0,NULL,&Text3,NULL,&spez3,3,NULL };

struct PropInfo prop2 =
{ AUTOKNOB|FREEVERT,0x0000,0x0000,0x0001,0x0001,
  NULL,NULL,NULL,NULL,NULL,NULL };

struct Gadget gdg2 =
{ &gdg3,270,38,20,90,NULL,RELVERIFY,PROPGADGET,
  &brd2,NULL,&Text2,NULL,&prop2,2,NULL };

struct StringInfo spez1 =
{ &DrwBuf[0],&UndoBuf[0],0,30,0,0,0,0,0,0,0,0,0 };

struct Gadget gdg1 =
{ &gdg2,85,20,190,10,NULL,RELVERIFY|TOGGLESELECT,STRGADGET,
  &brd0,NULL,&Text1,NULL,&spez1,1,NULL };



struct NewWindow NewWdw1 =
{
 200,60,306,180,       /*LeftEdge,TopEdge,Width,Height       */
 0,1,                  /*DetailPen,BlockPen                  */
 CLOSEWINDOW|GADGETUP,
 WINDOWDRAG|WINDOWDEPTH|      /*Bedeutung der Flags s.Text*/
 SMART_REFRESH|WINDOWCLOSE,
 &gdg1,NULL,           /*FirstGadget,CheckMark auf NULL      */
 "My FileRequester",   /*Titel des Windows                   */
 NULL,NULL,            /*Screen,BitMap auf NULL              */
 0,0,0,0,              /*Mini-und Maximale Groesse           */
 WBENCHSCREEN          /*Window auf WBenchScreen             */
};


void main()

{
        struct IntuiMessage *message;
        ULONG class;
        USHORT code, x, y, GadgetNr;
        int i;


         /* Als erstes die 10 Boolgadgets fuer die Erkennung, */
         /* welches File angeklickt wurde, initialisieren.    */
```

Bild 5. Nicht nur zur Demonstration öffnet dieser „File-Requester" zwanzig Gadgets. Das Programm darf so nur vom CLI aus gestartet werden

```
for(i=0;i<11;i++)
  {
   Gadg[i].NextGadget = &Gadg[i+1];
   Gadg[i].LeftEdge   = 20;
   Gadg[i].TopEdge    = 37+(i*8);
   Gadg[i].Width      = 225;
   Gadg[i].Height     = 8;
   Gadg[i].Flags      = GADGHCOMP;
   Gadg[i].Activation = RELVERIFY;
   Gadg[i].GadgetType = BOOLGADGET;
   Gadg[i].GadgetID   = 10+i;
  }


/* Links herstellen */

Gadg[10].NextGadget    = NULL;
gdg7.NextGadget        = &Gadg[0];


/* Erst Intuition.library oeffnen */

IntuitionBase = (struct IntuitionBase *)
     OpenLibrary("intuition.library",0);
        if(IntuitionBase==NULL) exit(200);


/* Dann Graphics.library oeffnen */

GfxBase= (struct GfxBase *)
        OpenLibrary("graphics.library",0);
        if(GfxBase==NULL) exit(200);



/* Jetzt das Window aufmachen */
/* Damit erscheinen auch gleich alle Gadgets */

wdw1=(struct Window *) OpenWindow(&NewWdw1);
if(wdw1==NULL) exit (50);


/* Auf Messages warten. Sobald eine ankommt,      */
/* abfragen welche Msg. ankam und reagieren.      */
/* Bei CLOSEWINDOW alles schliessen und beenden.  */
/* Bei GADGETUP je nach Gadget.Funktion ausloesen.*/


for(;;)
{
   Wait(1<<wdw1->UserPort->mp_SigBit);

   while(message=(struct IntuiMessage *)
                 GetMsg(wdw1->UserPort))

     {
      class=message->Class;  /* Parameter retten  */
      code= message->Code;
      x=    message->MouseX;
      y=    message->MouseY;
      GadgetNr= ((struct Gadget *)message->IAddress)->GadgetID;

      ReplyMsg(message);     /* Msg. zurueckgeben */
     }

   /* War es die CloseWindow-Message ? */

   if(class==CLOSEWINDOW||(class==GADGETUP&&GadgetNr==4))
     {
     CloseWindow(wdw1);
     CloseLibrary(IntuitionBase);
     exit(0);
     }

   /* oder war es es Gadget-Event ?   */

   if(class==GADGETUP)
     {
      switch(GadgetNr) /* In GadgetNr steht die Nummer */
                       /* die in dem Elemten GadgetID  */
                       /* angegeben wurde.             */
          {

           /* Wurde das obere StringGadget betaetigt ? */
           case 1:
             readdir(DrwBuf);
             printdir(FileCount);
             break;

           /* Wurde das PropGadget betaetigt ?         */
           case 2:
             printdir(FileCount);
             break;

           /* Wurde das untere StringGadget oder das  */
           /* "OK" Gadget betaetigt ?    Ja -> Ende   */
           case 3:
           case 7:
             printf("You Clicked:  %s\n",FileBuf);
             CloseWindow(wdw1);
             CloseLibrary(IntuitionBase);
             exit(0);

           /* Wurde das "DF1:" Gadget betaetigt ?     */
           case 5:
             strcpy(DrwBuf,"DF1:\x00");
             RefreshGadgets(&gdg1,wdw1,NULL);
             readdir(DrwBuf);
             printdir(FileCount);
             break;


           /* Wurde das "DF0:" Gadget betaetigt ?     */
           case 6:
             strcpy(DrwBuf,"DF0:\x00");
             RefreshGadgets(&gdg1,wdw1,NULL);
             readdir(DrwBuf);
             printdir(FileCount);
             break;


           /* Alle anderen Events muessen von den 11  */
           /* BoolGadgets zur Fileerkennung kommen !! */
           default:
             clicked((USHORT)(GadgetNr-10));
             break;
          }
     }
}
```

```
/* Die Funktion readdir liest das Directory, dessen String im */
/* Pointer path angegeben wurde,in die FILEINFOstrukturen ein */
/* und schreibt in die globale Variable FileCount die Anzahl  */
/* gelesenen Eintraege.                                       */

void readdir(pfad)

char *pfad;


{
        int i=0, error=0;
        char path[40];


        strcpy(path,pfad);        /* zuerst String kopieren */
        strcat(path,"/#?\x00");   /* dann /#? anhaengen     */


        dfind(&info[0],path,0);   /* ersten Eintrag finden  */

        i=0;                      /* weitere Eintraege lesen*/
        do
          {
           oldinfo=info[i];
           error=dnext(&oldinfo);    /* bis error != 0      */
           info[i+1]=oldinfo;
           i=i+1;
          }
        while(error==0);

        FileCount=i-1;
}


/* Diese Routine schreibt unter Beruecksichtigung des PropGadgets*/
/* i Filenamen aus den FILEINFOstrukturen in das Window. Vorher  */
/* muss readdir() aufgerufen worden sein, da ansonsten der Inhalt*/
/* der FILEINFOstrukturen nicht definiert ist.                   */


void printdir(i)
int i;

{
        int c;


        prop2.VertBody = MAXBODY;
        Offset=0;

        if(i>10)    /* mehr als 10 Files ? */
           {
            prop2.VertBody = MAXBODY/i*10;
            Offset=prop2.VertPot/(MAXBODY/(i-11));
           }
        RefreshGadgets(&gdg1,wdw1,NULL);
        SetAPen(wdw1->RPort,0);    /* Die "alten" Files auf dem */
        SetBPen(wdw1->RPort,0);    /* Window loeschen           */
        SetDrMd(wdw1->RPort,1);
        RectFill(wdw1->RPort,20,37,261,125);


        /* und die neuen schreiben. */

        for(c=Offset;c<i&&(c<Offset+11);c++)
          {
           IText.IText = info[c].fib_FileName;
           PrintIText(wdw1->RPort,&IText,20,37+8*(c-Offset));
          }
}


/* Hier wird der Inhalt der "angeklickten" FILEINFOstruktur in */
/* den Puffer des "File:" Gadgets kopiert.                     */


void clicked(i)
USHORT i;


{
        strcpy(FileBuf,info[Offset+i].fib_FileName);
        RefreshGadgets(&gdg1,wdw1,NULL);
}
```

linken (oberen) Anschlag, so ist der Wert Null, am rechten (unteren) Anschlag $FFFF. Man kann mit diesen Elementen auch den Startwert eines Gadgets setzen, näheres kann dem Programm entnommen werden.

HorizBody, VertBody
Diese Elemente geben an, welchen Prozentsatz der Gesamtinformation das Gadget anzeigt. Muß man z. B. 15 Zeilen Text anzeigen und hat nur 5 Zeilen Platz, so werden die Variablen auf ⅓ gestellt.

Die übrigen Elemente der PropInfo-Struktur werden von Intuition initialisiert.

Das Programm

Im Gegensatz zum Programm des ersten Teils, das mehr erklärenden Charkter hatte, gibt es heute ein praxisorientiertes Beispiel. Die meisten Amiga-Benutzer werden die sogenannten „Filerequester" kennen. Der hier vorgestellte ist recht brauchbar, da er ohne Probleme in eigene Programme eingebaut werden kann. Auf die Directory-Auswahl wurde aus Gründen der Übersichtlichkeit verzichtet. Das Programm läuft mit dem Lattice-C-Compiler Version 3.1 und erzeugt immerhin 20 (!) Gadgets, womit es sich hervorragend als Demonstration für diese Thematik eignet. Wer auch noch die Feinheiten wissen will, kommt um eine nähere Betrachtung von *Bild 5* nicht herum.
Zum Abschluß noch einige Erweiterungsvorschläge: Wer Lust hat, kann diesen Requester so umschreiben, daß er auch Directories verwalten kann. Dazu muß in der Funktion readdir(pfad) die Zeile
dfind(&info[0],path,0)
in die geändert Form
dfind(&info[0],path,–1)
gebracht werden. Dann liest readdir auch Directories. Damit ist es aber nicht getan: Diese Directories müssen natürlich auch noch verwaltet werden...
Eine andere Möglichkeit ist, beim PropGadget die Dateinamen auch während des „Bewegens" des Knopfes anzuzeigen. Ein Tip: Zusätzlich zu RELVERIFY auch noch GADGIMMEDIATE in der Gadgetstruktur setzen und GADGETUP in der NewWindowstruktur. Das Programm erhält dann schon beim Anklicken des Rollbalkens eine Message und zeigt bis zum Eintreffen der GADGETUP-Message das Directory mit printdirc) immer wieder an.
Soweit für heute: Beim nächsten (und letzten) Mal werden wir uns mit Menüs auseinandersetzen. Viel Spaß einstweilen mit den Gadgets! □

Ingo Eickmann

Editieren der EGA-Palette

Bei fast allen Programmen werden die 16 darstellbaren Farben so belassen, wie man sie aus dem Textmode der CGA-Karte kennt. Es stehen jedoch insgesamt 64 Farbtöne zur Verfügung, mit denen man nach Belieben völlig neue Paletten zusammenstellen kann.

Die 16 gleichzeitig darstellbaren Farben können entweder über das Palettenregister (Port 3C0H, Indizes 0...F) oder über den Interrupt 10H verändert werden, der hier benutzt wird. Über die zweite Möglichkeit kann man übrigens auch von Turbo-Pascal aus schnell Änderungen vornehmen.
Folgende Parameter müssen beim Aufruf des INT 10H übergeben werden:

ah = 10H Setzen der Palettenregister

al = 0 Setzen der Farbpalettenregister
bl Registernummer (00...0F)
bh Farbwert (00...3F)

al = 1 Setzen des Overscanregisters
bh Farbwert (00...3F)

al = 2 Setzen aller Farbregister
ES:DX Vektor auf Wertetabelle:
<Reg01><Reg02>..<Reg0F>,
<Overscan>

Das Bitmuster des Farbwertes setzt sich so zusammen:

Bit 0: blau	Bit 4: hellgrün
Bit 1: grün	Bit 5: hellrot
Bit 2: rot	Bit 6: nicht verwendet
Bit 3: hellblau	Bit 7: nicht verwendet

Das abgedruckte Programm (*Bild*) kann mit oder ohne Parameter aufgerufen werden. Ohne Parameter initialisiert es den 80-Zeilen-Bildschirm, bringt alle 16 Farben der EGA-Karte in einem Balken auf den Bildschirm und gibt in der Zeile darunter die dazugehörigen Farbwerte aus. Mit Hilfe der Cursor-Tasten lassen sich nun die einzelnen Register anwählen und beliebig verändern. Mit <ENTER> wird zurück ins DOS gesprungen, die eingestellten Werte bleiben erhalten.
Will man eine bestimmte Palette direkt aktivieren, dann genügt das Anfügen der Farbwerte an den Programmnamen, jeweils durch Kommas getrennt:

PALETTE
<Reg01>,<Reg02>,...,<Reg0F>

Gelbe Schrift auf blauem Hintergrund bei sonst unveränderter Palette erhält man also durch:

PALETTE 1,,,,,,,7

Der Aufruf kann auch vor einem Turbo-Pascal-Programm erfolgen, da Turbo-Pascal die Farben selbst nicht modifiziert.

Durch die Wahl von ausschließlich Blautönen (1, 3, 9, 11, 15, 17, 25, 27, 31, 33, 41, 43, 49, 57) oder Tönen einer anderen Grundfarbe kann man bei 3D-Grafiken Oberflächen der Reflexion entsprechend aufhellen oder verdunkeln. Bilder, die mit dem Professional-Grafik-Adapter in einer Grundfarbe erzeugt wurden, können durch eine Umwandlung in die passenden Farbwerte der EGA-Karte gut dargestellt werden. □

```
                                ; Definition der EGA-Farbpalette
                                ; I.Eickmann, §1987
                                ;
                                ; Aufruf des Editors durch PALETTE<0D>
                                ; Ändern einzelner Register durch
                                ;    PALETTE <Reg0>,<Reg1>,...<Reg0F>
                                ;
0000                            code segment
                                        assume cs:code, ds:code
0100                                    org 100H
0100  33 DB                     start:  xor bx,bx               ;Einsprung
0102  33 C9                             xor cx,cx
0104  BF 0080                           mov di,80H
0107  8A 0D                             mov cl,byte ptr [di]
0109  80 F9 00                          cmp cl,0                ;Parameter?
010C  75 03                             jnz uebergabe           ;ja   => Ändern
010E  E9 0193                           jmp normal              ;nein => Editor

0111  47                        uebergabe: inc di               ;Einlesen der
0112  8A 05                             mov al,[di]            ;Übergabepara-
0114  3C 20                             cmp al,' '             ; meter
0116  74 69                             jz uebergabe1
0118  3C 2C                             cmp al,','             ;Trennzeichen?
011A  75 31                             jnz uebergabe2
011C  FE C7                             inc bh
011E  80 FF 0F                          cmp bh,15              ;Setzen des
0121  7F 05                             jg fehler              ;letzten Registers?
0123  32 DB                             xor bl,bl
0125  EB 5A 90                          jmp uebergabe1

0128  0E                        fehler: push cs                ;Fehlermeldung
0129  1F                               pop ds                 ; ausgeben
012A  BA 0134                           mov dx,offset fehler_message
012D  B4 09                             mov ah,9
012F  CD 21                             int 21H
0131  EB 5C 90                          jmp dos
0134  21 21 20 45 52 52         fehler_message: db '!! ERROR on input !!',0AH,
                                0DH,'$'

014D  3C 30                     uebergabe2: cmp al,48          ;Eingabe im
014F  7C D7                             jl fehler              ;Zahlenbereich?
0151  3C 39                             cmp al,57
0153  7F D3                             jg fehler
0155  2C 30                             sub al,48              ;ASCII => Wert
0157  8A D0                             mov dl,al
0159  51                               push cx
015A  33 C9                             xor cx,cx
015C  8A CF                             mov cl,bh              ;Ermitteln der
015E  BD 01D6                           mov bp,offset para     ;Bufferaddresse
0161  03 E9                             add bp,cx
0163  8A 46 00                          mov al,[bp]            ;Laden der Zehner-
0166  FE C3                             inc bl                 ; stelle
0168  59                                pop cx
0169  80 FB 02                          cmp bl,2               ;Zehner vorhanden?
016C  74 04                             jz zehner
016E  7F B8                             jg fehler              ;Hunderter?
0170  32 C0                             xor al,al
0172  B6 0A                     zehner: mov dh,10              ;Berechnen des
0174  F6 E6                             mul dh                 ;Registerinhalts
0176  72 B0                             jc fehler              ;aus Zehner und
0178  02 C2                             add al,dl              ;aktuellem Wert
017A  3C 40                             cmp al,64
017C  7F AA                             jg fehler              ; >64 => ERROR
017E  88 46 00                          mov byte ptr [bp],al
0181  E2 8E                     uebergabe1: loop uebergabe
```

Die Farbendarstellung der EGA-Karte läßt sich mit diesem Programm manipulieren

```
0183  0E                    push cs
0184  1F                    pop ds
0185  BA 023A               mov dx,offset message1
0188  B4 09                 mov ah,9                ;Meldung ausgeben
018A  CD 21                 int 21H
018C  E8 0252               call auspal             ;neue Palette
018F  B4 4C           dos:  mov ah,4CH              ;setzen, ins DOS
0191  CD 21                 int 21H

0193  B8 0003         normal: mov ax,3              ;EDITOR
0196  CD 10                 int 10H
0198  0E                    push cs
0199  1F                    pop ds
019A  BA 01E7               mov dx,offset message
019D  B4 09                 mov ah,9
019F  CD 21                 int 21H                 ;Meldung ausgeben
01A1  32 DB                 xor bl,bl
01A3  B9 0005         color_aus: mov cx,5           ;Farbbalken
01A6  B8 09DB               mov ax,9DBH             ; ausgeben
01A9  CD 10                 int 10H
01AB  53                    push bx
01AC  B4 03                 mov ah,3
01AE  32 FF                 xor bh,bh
01B0  CD 10                 int 10H
01B2  32 FF                 xor bh,bh
01B4  B0 05                 mov al,5
01B6  02 D0                 add dl,al
01B8  B4 02                 mov ah,2
01BA  CD 10                 int 10H
01BC  5B                    pop bx
01BD  FE C3                 inc bl
01BF  80 FB 10              cmp bl,16
01C2  7C DF                 jl color_aus
01C4  0E                    push cs
01C5  1F                    pop ds
01C6  E8 0276               call pal_init           ;Palette neu
01C9  E8 028C               call zahlaus            ;editieren
01CC  BA 024F               mov dx,offset message2
01CF  B4 09                 mov ah,9                ;<CR> ausgeben
01D1  CD 21                 int 21H
01D3  EB BA                 jmp dos                 ;Rücksprung

01D5  00              akt     db 0
01D6  FF FF FF        para    db 17 dup(0FFH)
01E7  0A 0D 50 41     message db 0AH,0DH,'PALETTE <Reg00>,<Reg01>,...
      4C 45 54        .,<Reg0F>                              by I.Eickmann
                      , ©1987'
0239  24                      db '$'
023A  62 79 20 49     message1: db 'by I.Eickmann, ©1987$'
      2E 45 69
024F  0D 0A 24        message2: db 0DH,0AH,'$'

0252                  auspal  proc near             ;Übergabe der
0252  B9 0010                 mov cx,16               ;neuen Werte
0255  BD 01E6         ausgabe: mov bp,offset para + 16
0258  2B E9                   sub bp,cx             ;für die Farb-
025A  51                      push cx               ; register
025B  B0 10                   mov al,16
025D  2A C1                   sub al,cl
025F  8A D8                   mov bl,al
0261  B8 1000                 mov ax,1000H
0264  8A 7E 00                mov bh,byte ptr [bp]
0267  80 FF FF                cmp bh,0FFH           ;bei <0FFH>
026A  74 06                   jz nicht_auspal       ; nichts ausge-
026C  33 C9                   xor cx,cx             ; ben
026E  33 D2                   xor dx,dx
0270  CD 10                   int 10H
0272  59              nicht_auspal: pop cx
0273  E2 E0                   loop ausgabe
0275  C3                     ret
                      auspal  endp

0276                  pal_init proc near            ;Werte einer
0276  33 C0                   xor ax,ax             ; initialisier-
0278  BD 01D6                 mov bp,offset para    ; ten EGA-Karte
027B  88 46 00        pal_i2:  mov byte ptr [bp],al ;in den Buffer
027E  FE C0                    inc al                ; schreiben
0280  45                       inc bp
0281  3C 08                   cmp al,8
0283  7C 02                   jl pal_i1
0285  0C 30                   or al,48
0287  3C 40           pal_i1:  cmp al,64
0289  7C F0                   jl pal_i2
028B  C3                       ret
                      pal_init endp

028C                  zahlaus  proc near            ;Farbwerte
028C  BD 0000                 mov bp,0              ; unter den
028F  B4 02                   mov ah,2              ; Farbbalken
0291  32 FF                   xor bh,bh             ; schreiben
0293  BA 0300                 mov dx,0300H
0296  CD 10                   int 10H
0298  BE 01D6                 mov si,offset para
029B  B3 0F           zahlaus1: mov bl,15           ;Video-Attribut
029D  B0 20                   mov al,20H            ;Cursor
029F  E8 0343                 call aus_z            ; positionieren
02A2  0B ED                   or bp,bp
02A4  74 06                   jz erste
02A6  E8 0343                 call aus_z
02A9  E8 0343                 call aus_z
02AC  3E: 8A 02       erste:  mov al,byte ptr ds:[si+bp]
02AF  32 E4                   xor ah,ah             ;Farbwert auslesen
02B1  33 D2                   xor dx,dx             ; und in ASCII-
02B3  B2 0A                   mov dl,10             ; String umwandeln
02B5  F6 F2                   div dl
02B7  50                      push ax
02B8  04 30                   add al,30H
02BA  8A 16 01D5              mov dl,akt
02BE  3B D5                   cmp dx,bp
02C0  75 03                   jnz zahlaus8
02C2  80 CB 80                or bl,80H             ;Blinken beim
02C5  E8 0343         zahlaus8: call aus_z          ; aktuellen
02C8  58                      pop ax                ; Register
02C9  B0 30                   mov al,30H            ; und Ausgabe
02CB  02 C4                   add al,ah
02CD  E8 0343                 call aus_z
02D0  45                       inc bp
02D1  83 FD 10                 cmp bp,16             ;letztes
02D4  7C C5                    jl zahlaus1           ; Register?

02D6  B8 0000         zahlaus2: mov ax,0            ;Keyboard
02D9  CD 16                    int 16H               ; abfragen
02DB  33 DB                    xor bx,bx
02DD  8A 1E 01D5               mov bl,akt
02E1  8B EB                    mov bp,bx
02E3  80 FC 4D                 cmp ah,4DH            ;rechts?
02E6  75 11                    jnz zahlaus9
02E8  A0 01D5                  mov al,akt            ;aktuelles
02EB  FE C0                    inc al                ; Register
02ED  3C 10                    cmp al,16             ; ändern
02EF  7C 02                    jl zahlaus4
02F1  32 C0                    xor al,al
02F3  A2 01D5         zahlaus4: mov akt,al
02F6  EB 40 90                 jmp zahlaus3
02F9  80 FC 4B        zahlaus9: cmp ah,4BH          ;links?
02FC  75 11                    jnz zahlaus10
02FE  A0 01D5                  mov al,akt            ;aktuelles
0301  FE C8                    dec al                ; Register
0303  3C 00                    cmp al,0              ; ändern
0305  7D 02                    jge zahlaus5
0307  B0 0F                    mov al,15
0309  A2 01D5         zahlaus5: mov akt,al
030C  EB 2A 90                 jmp zahlaus3
030F  80 FC 50        zahlaus10: cmp ah,50H         ;runter?
0312  75 11                    jnz zahlaus11
0314  3E: 8A 02                mov al,byte ptr ds:[si+bp]
0317  FE C0                    inc al
0319  3C 40                    cmp al,64             ;Farbwert
031B  75 02                    jnz zahlaus6          ; ändern
031D  32 C0                    xor al,al
031F  3E: 88 02       zahlaus6: mov byte ptr ds:[si+bp],al
0322  EB 14 90                 jmp zahlaus3
0325  80 FC 48        zahlaus11: cmp ah,48H         ;hoch?
0328  75 14                    jnz zahlaus12
032A  3E: 8A 02                mov al,byte ptr ds:[si+bp]
032D  FE C8                    dec al
032F  3C FF                    cmp al,0FFH           ;Farbwert
0331  75 02                    jnz zahlaus7          ; ändern
0333  B0 3F                    mov al,63
0335  3E: 88 02       zahlaus7: mov byte ptr ds:[si+bp],al
0338  E8 0252         zahlaus3: call auspal         ;Palette
033B  E9 028C                  jmp zahlaus           ; setzen
033E  3C 0D           zahlaus12: cmp al,0DH
0340  75 94                    jnz zahlaus2          ;Return bei
0342  C3                       ret                   ; <ENTER>
                      zahlaus  endp

0343                  aus_z    proc near            ;ASCII in al
0343  50                      push ax               ; mit Attribut
0344  32 FF                   xor bh,bh             ; in bl
0346  53                      push bx               ; ausgeben und
0347  B4 03                   mov ah,3              ; Cursor
0349  CD 10                   int 10H               ; verschieben
034B  FE C2                   inc dl
034D  B4 02                   mov ah,2
034F  CD 10                   int 10H
0351  5B                      pop bx
0352  58                      pop ax
0353  B4 09                   mov ah,9
0355  B9 0001                 mov cx,1
0358  CD 10                   int 10H
035A  C3                      ret
                      aus_z    endp

035B                          code ends
                              end start
00000 ERRORS
```

Raul Rojas

Vorführeffekt ade

Die Simulation von Tastatureingaben

Pannen bei Software-Demonstrationen sollen ausgeschlossen sein. Mit diesem Assembler-Programm scheint für einen nichtsahnenden Beobachter ein Gespenst jede Sekunde eine Taste zu drücken. Das Listing verschafft grundlegende Einsichten über die Vorgänge bei der Tastaturbenutzung.

Jedes Anwenderprogramm kann die von diesem Programm in den Tastaturpuffer geschobenen Zeichen lesen. So ist es zum Beispiel möglich, MS-Word aufzurufen, eine Datei mit MS-Word zu laden und sie mit den Cursortasten zu inspizieren, alles per Simulation.

Das vorliegende Programm kann auch benutzt werden, um die Funktionsweise von Batch-Dateien zu zeigen oder um die Arbeitsweise von Netzwerken vorzuführen, ohne daß dafür Personen tatsächlich an den einzelnen Arbeitsplätzen sitzen müssen. Für Computerläden ist es interessant, Software, die über keine Demo-Option verfügt, auch nachts im Schaufenster automatisch laufen lassen zu können.

Nur einzelne Anwender-Programme haben derartige Simulationen bereits vorgesehen. Superkey und die neue Version von Sidekick (arbeitet jedoch nicht einwandfrei mit deutscher Tastatur) sind dazuzurechnen, auch das Textverarbeitungssystem Euroscript (mit der Datei STARTER.INT) und das Datenbanksystem Paradox (mit Scripts) bieten Ähnliches.

Der Tastatur-Puffer

Der IBM-PC und Kompatible lesen die Tastatur nicht durch Polling, sondern mittels des Interrupt 9. Jedesmal wenn eine Taste gedrückt wird, erzeugt die Tastatur einen Hardware-Interrupt. Der Prozessor kümmert sich dann um den Interrupt, liest von der Tastatur den Scan-Code der gedrückten Taste und speichert die ASCII- und Scan-Codes der Taste in einem speziellen Puffer mit der Adresse 040H:01EH ab. In dem Puffer können bis zu 16 Zeichen gespeichert werden. Wenn der Puffer voll ist (weil zu schnell getippt wurde oder weil das Programm noch keine Zeichen gelesen hat), erzeugt der Code für Interrupt 9 einfach ein Beep und löscht den Interrupt.

Der Tastatur-Puffer ist als Ring angelegt. Zwei Zeiger klammern die noch nicht gelesenen Zeichen aus: der Kopf-Zeiger (Adresse 040H:01AH) avanciert zwei Byte mit jedem von Interrupt 9 gespeicherten Zeichen. Der Tail-Zeiger folgt dem Kopf und avanciert zwei Byte, jedesmal wenn das Programm ein Zeichen liest (womit zwei Byte aus dem Tastaturpuffer entfernt werden). Zeigen Kopf und Tail zur selben Adresse, so ist der Puffer leer. Falls einer der Zeiger bis zum Ende des Puffers gelangt, so wird der Zeiger bis zum Anfang des Puffers zurückgebracht (Wrapping).

Das hier vorgestellte Programm leitet den Interrupt 9 um. Falls es aktiv ist, wird kein Zeichen von der Tastatur in den Tastaturpuffer gebracht. Das Zeichen geht verloren, und das Programm kann ungestört weitertippen. Falls das Zeichen das Minus des

```
vector     struc                   ;Definition einer Struktur
           regip dw ?              ;
           regcs dw ?              ;
vector     ends                    ;

code       segment
           assume    cs:code,ds:code,es:code,ss:code
           int 020h                ;Program-Segment-Prefix
top_mem    dw   ?                  ;
           db   ?
           db    5 dup(?)
terminate vector <>
break      vector <>
error      vector <>
           dw   ?
           db   20 dup(?)
environment    dw   ?
           dw    7 dup(?)
           db   20 dup(?)
dos_call   proc far
           int  021h
           ret
dos_call   endp
           db    9 dup(?)
area_1     db   16 dup(?)
area_2     db   16 dup (?)
           db    4 dup (?)
lenght     db    7
area_3     db  127 dup (?)
;
ghost      label near              ;
           jmp start               ;Installation
           assume    cs:code,ds:nothing,es:nothing,ss:nothing
           text      db 0dh,0ah,"GHOST WRITER",0d,0ah,"$"
           orglc     vector <>
           org9      vector <>
           counter   db 0       ;Zähler
           aktiv     db 1       ;Aktiv-Flag
           zeige     dw 0h      ;Pointer zur Zeichentabelle

tchl db 100d,00h                   ;d
     db 105d,00h                   ;i
     db 114d,00h                   ;r
     db 13d,28d                    ;<ret>
     db 01h,01h                    ;5 Sek. Pause
     db 01h,01h                    ;
     db 01h,01h                    ;
     db 01h,01h                    ;
     db 01h,01h                    ;
     db 00h,00h
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
keybo      proc far                ;Neuer Tastatur-Interrupt
           push ax                 ;rette Register
           push bx                 ;
           in al,060h              ;lese Tastatur
           mov bl,al               ;rette Scan-Code in bl
           in al,061h              ;lese Kontrollbyte
           or al,082h              ;
           out 061h,al             ;reset Kontrollbyte
           and al,07fh             ;
           out 061h,al             ;reset Kontrollbyte
           mov al,020h             ;
           out 020h,al             ;reset Interrupt
           mov al,aktiv            ;
```

Assemblerprogramm zur Simulation von Tastatureingaben

Cursor-Blocks ist, wird gestoppt. Der Computer verhält sich wieder normal. Mit der Plus-Taste des Cursor-Blocks geht die Tipperei wieder von vorne los.

Der Interrupt 1CH

Für die kontrollierte Eingabe der Zeichen wird der bekannte und immer wieder mißhandelte Interrupt 01CH benutzt. Dieser Interrupt wird auf den residenten Teil des Programms umgeleitet. Die Anzahl der Interrupts wird gezählt, und zum Tippen eines neuen Zeichens wird der ASCII- und Scan-Code des Zeichens aus einer Tabelle geholt und in den Tastatur-Puffer gespeichert. Es wird immer überprüft, ob der Tastaturpuffer schon leer ist oder ein Zeiger das Ende des Puffers erreicht hat. Ein Tastenklick mit Hilfe der PC-Lautsprecher macht das Programm noch realistischer (einige sagen entnervender).

Der ASCII- und Scan-Code 01H wird im Programm als eine Pause interpretiert. Ein Null-Wort schließt die Tabelle ab.

Verbesserungen möglich

Das Programm arbeitet einwandfrei auf IBM-PC- und kompatiblen Rechnern. Es ist denkbar, ein paar Verbesserungen einzubauen, zum Beispiel eine Record-Funktion. Durch die Umleitung des Interrupt 9 könnte damit im normalen Betrieb jeder Tastendruck (in der Ghost-Tabelle) gespeichert und später durch eine Play-Option wiederholt werden. Dies wäre ohne großen Aufwand durchführbar, und es vereinfacht den Aufbau der Code-Tabelle. Für den Aufbau der Tabelle kann man auch ein Pascal-Programm benutzen, das aus einer Datei die Zeichen- und Scan-Codes liest und die Datei ghost.asm generiert. Diese kann dann compiliert werden.

Im Listing ist ab dem Label tch1 als Beispiel eine ständige Wiederholung des Befehls DIR vorgesehen. Bei manchen MSDOS-Rechnern (zum Beispiel dem Olivetti) sind zur Erkennung der Tasten – und + (zum Abbruch und Neustart des Programmes) Änderungen im Bereich hinter dem Label keybo erforderlich. Dort gehen die Scancodes der beiden Tasten ein.

Flugsimulator um Mitternacht

Eine effektvolle Erweiterung wäre das Erwachen des Programmes zu bestimmten Zeiten. So könnte einem als Nachtarbeiter bekannten Kollegen ein gehöriger Schreck in die Glieder fahren, wenn sein Rechner pünktlich zu Mitternacht anfängt, Flugsimulator zu spielen. □

```
          or al,al             ;teste Aktiv-Flag
          jz notaktiv          ;springe falls Null
          cmp bl,04ah          ;siehe ob "-"
          je aus               ;wenn ja, ausschalten
          jmp retint           ;return
aus:      mov aktiv,0          ;setze Aktiv-Flag=0
          mov zeige,0          ;setze Zeiger=0
          jmp retint           ;return
notaktiv: cmp bl,04eh          ;siehe ob "+"
          je einschal          ;wenn ja, einschalten
          jmp done             ;return über alte Interrupt
einschal: mov aktiv,1          ;setze Aktiv-Flag:=1
retint:   pop bx               ;Register wieder laden
          pop ax               ;
          iret                 ;return from interrupt
done:     pop bx               ;Register wieder laden
          pop ax
          jmp org9             ;springe zum alten Interrupt
keybo     endp
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
typer     proc far             ;Neuer Interrupt 1CH
          push ax              ;rette Register
          push bx              ;
          push cx              ;
          push ds              ;
          mov al,aktiv         ;teste Aktiv-Flag
          or al,al             ;
          jz trick             ;falls aus, return
          mov al,counter       ;siehe counter
          inc al               ;
          cmp al,10            ;wenn gleich 10, type etwas
          jz sigue             ;weiter machen
          mov counter,al       ;sonst speichere neuen Counter
trick:    jmp raus             ;return

sigue:    mov counter,0        ;reset Counter
          mov ax,040h          ;
          mov ds,ax            ;DS=040H
          mov bx,lah           ;
          mov al,[bx]          ;lese Head-Stellung
          xor cx,cx            ;
          mov bx,lch           ;
          mov cl,[bx]          ;lese Tail-Stellung
          cmp al,cl            ;vergleiche beide
          jne raus             ;return wenn beide gleiche sind
          mov bx,zeige         ;
          mov ah,byte ptr tchl[bx] ;lese sonst einen ASCII-Code
          inc bx               ;
          mov al,byte ptr tchl[bx] ;und einen Scan-Code
          or ax,ax             ;teste ob am Ende
          jne weiterso         ;wenn nicht, mache weiter
          mov zeige,ax         ;falls am Ende, wrap
          jmp raus             ;und return
weiterso: cmp ax,0101h         ;siehe ob Pause
          jne doit             ;wenn nicht, tippe
          mov ax,zeige         ;wenn ja, überspringe Pause
          inc ax               ;
          inc ax               ;
          mov zeige,ax         ;
          jmp raus             ;und return
doit:     mov bx,cx            ;
          mov byte ptr [bx],ah;neues ASCII in Tail
          inc bx               ;
          mov byte ptr [bx],al;neues Scan-Code auch
          inc bx               ;
          mov ax,03h           ;
          out 061h,al          ;mache einen kleinen Klick
          mov cx,0400h         ;
dort:     mov ax,09h           ;und warte
          loop dort            ;
          xor ax,ax            ;reset
          out 061h,al          ;Lautsprecher
          mov ax,zeige         ;lade zeige
          inc ax               ;
          inc ax               ;
          mov zeige,ax         ;zeige:=zeige+2
          mov ax,bx            ;
          cmp ax,03eh          ;am Ende des Buffers?
          jb non               ;
          mov ax,01eh          ;Falls ja, wrap
non:      mov bx,01ch          ;
          mov [bx],ax          ;speichere neue Tail
raus:     pop ds               ;Register neu laden
          pop cx               ;
          pop bx               ;
          pop ax               ;
          jmp orglc            ;mit alten Interrupt verketten
typer     endp                 ;
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
start:    mov ax,2000h         ;setze neue
          out 042h,al          ;Frequenz des Lautsprechers ein
          mov al,ah            ;
          out 042h,al          ;
          xor ax,ax            ;
          mov zeige,ax         ;zeige:=0
          mov dx,offset text   ;
          mov ah,09h           ;
          int 21h              ;schreibe Startmeldung
          push es;             ;rette ES
          mov ah,035h          ;
          mov al,01ch          ;
          int 021h             ;bringe altes Interrupt 01CH
          mov orglc.regip,bx   ;rette Adresse
          mov orglc.regcs,es   ;
          mov ah,035h          ;
          mov al,09h           ;bringe altes Interrupt 09H
          int 021h             ;
          mov org9.regip,bx    ;rette Adresse
          mov org9.regcs,es    ;
          pop es               ;ES rücksetzen
          mov ah,025h          ;
          mov al,01ch          ;
          mov dx,offset typer  ;setze neues Interrupt 01CH
          int 021h             ;
          mov ah,025h          ;
          mov al,09h           ;
          mov dx,offset keybo  ;setze neue Tastatur-Interrupt
          int 021h             ;
          push es              ;
          mov ah,049h          ;
          mov es,environment   ;mache Speicher frei
          int 021h             ;
          pop es               ;
          mov dx,offset start  ;
          int 027h             ;Ende, aber bleibe resident
code      ends                 ;
end       ghost
```

Peter Wollschlaeger

C-Funktionen im Amiga-ROM

... und wie man sie in Assembler einsetzt

Normalerweise bindet man Assembler-Routinen in C-Programme ein. Beim Amiga ist der umgekehrte Weg empfehlenswert, denn einige nützliche C-Funktionen befinden sich da im ROM. Wir zeigen, wie man ganz legal auf diese Routinen zugreifen und damit Programmierarbeit sparen kann.

Wenn man weiß, wie man das „ROM Kernel Reference Manual" von Commodore lesen muß, entdeckt man interessante Dinge. Die Lesevorschrift: Alles, was groß und fett gedruckt ist, ist trivial, die interessanten Dinge stehen kleingedruckt im Anhang. So findet man auf Seite B12 von Exec ein paar Seiten mit dem Titel „amiga.lib.doc". Diese Seiten enthalten den Hinweis, daß unter anderem die C-Funktion printf() im ROM vorhanden ist. Daher empfiehlt Commodore C-Programmierern, „amiga.lib" vor die Laufzeit-Bibliothek des C-Compilers zu linken, damit der Code kürzer wird. mc empfiehlt, die Routinen in Assembler zu nutzen, denn das spart so aufwendige Übungen wie die Umwandlung von Binärzahlen in hexadezimale oder dezimale Zeichenketten.

```
        opt     l+
* Linken Sie diesen File als xxx.o mit
* blink AStartup.obj,xxx.o to xxx library lib:amiga.lib

* Demo ROM-printf() in Assembler
        XREF.L  _printf         ;.L nur beim DevPac
        XDEF    _main
_main
        pea     333             ;Zahl 3
        pea     222             ;Zahl 2
        pea     111             ;Zahl 1
        pea     fmt             ;Format-String
        jsr     _printf
        lea     16(sp),sp       ;4 Longs vom Stack
        rts

fmt     dc.b    'Z1 = %ld  Z2 = %ld  Z3 = %ld',10,0
```

Bild 1. Am Beispiel von printf(): So erfolgt der Aufruf der ROM-Funktionen

Die Routinen sind für den Aufruf von C her vorgesehen, erwarten also ihre Parameter auf dem Stack. Folglich muß man in Assembler seine Parameter auf den Stack bringen, dann die Funktion mit JSR aufrufen und schließlich den Stackpointer wieder auf den alten Wert bringen. Ansonsten muß man noch bedenken, daß C die Parameter einer Funktion von rechts nach links liest, daß heißt aus dem C-Aufruf

```
function (a, b, c);
```

wird in Assembler

```
pea c
pea b
pea a
jsr function
```

Um nun die – in diesem Beispiel 3 – Langworte (= 12 Bytes) wieder vom Stack zu holen, ist dann noch der Befehl

```
lea 12(sp),sp
```

erforderlich. Die praktische Anwendung davon zeigt *Bild 1* am Beispiel der Funktion printf(). Nach den drei Daten muß hier noch die Adresse einer Format-Zeichenkette übergeben werden. Beachten Sie ein paar feine Unterschiede zur C-Schreibweise und zum Standard-C der Compiler-Bibliotheks-Funktionen. Das C-Äquivalent von Bild 1 hieße:

```
printf („Z1 = %ld Z2 = %ld Z3 = %ld\n",
        111, 222, 333);
```

In der Format-Zeichenkette wird nur das Escape-Zeichen „%" nebst Zubehör erkannt, der Rest gilt als Text. So ist zum Beispiel „\n" nicht möglich, kann aber einfach durch ein Byte mit dem dezimalen Wert 10 ersetzt werden. Die Zeichenkette kann also beliebig zusammengesetzt werden, als Abschluß gilt das Null-Byte.

Das ROM-printf() kann nur ganze Zahlen, Zeichen und Zeichenketten formatieren. Wenn man die Routine von C aus aufruft, ist für Zahlen nur „%l" (= Langwort) erlaubt. Nur in Assembler können auch Worte (jedoch keine Bytes, da mit dem Stack gearbeitet wird) übergeben werden.

Es gibt – hier am Beispiel %l – folgende Varianten:

%lc Das nächste Langwort wird als Zeichen (8 Bit) ausgegeben

%ld Ausgabe als Dezimalzahl

%lx Ausgabe als Hexadezimalzahl

%ls Ausgabe als Zeichenkette (Übergeben wird die Adresse einer mit Null abgeschlossenen Zeichenkette)

Dem „%" können noch diese Zeichen folgen:

– (Minus): Innerhalb der Feldweite wird links justiert

nn Eine Zahl für die Feldweite, Auffüllen mit Leerzeichen

0 führende Nullen werden nicht durch Leerzeichen ersetzt

.nn Eine Zahl nach dem Punkt für das Maximum an Zeichen

snn Eine Zahl nach %s für die maximale Anzahl Zeichen, die als Zeichenkette ausgegeben werden sollen

Das Programm von Bild 1 wurde mit dem DevPac-Assembler geschrieben. Beim Amiga-Assembler muß die Opt-Direktive weglassen und XREF anstatt XREF.L geschrieben werden. Die Funktion printf() verlangt einige Referenzen zur Ein-/Ausgabe, die im Startup-Modul definiert sind. Deshalb muß man „AStartup.obj" und natürlich auch „amiga.lib" beim Linker-Lauf mit einbinden. Vergessen Sie nicht, anstatt „lib:" ggf. Ihren Pfadnamen für „amiga.lib" einzusetzen.

Das ausführbare Programm belegt rund 2,5 KByte, was zwar schon weniger als die 7,5 KByte des C-Pendants ist, doch immer noch recht viel: Der größte Teil geht für das Startup-Modul drauf. Deshalb zeigt *Bild 2*,

```
        opt     l+                          ;linkbarer Code
*-------------------------------------------------------
* Diesen File xxx.o können Sie ohne AStartup.obj linken mit
* blink xxx.o to xxx library lib:amiga.lib
*-------------------------------------------------------
* Demo printf() und weitere in einem Console-Window
* ------------------------------------------------
        incdir  ":devpac/include/"
        include exec/exec_lib.i
        include libraries/dos_lib.i
        include libraries/dos.i

        XREF.L  _printf                 ;importieren wir
        XREF.L  _RangeRand
        XREF.L  _getchar

        XDEF    _DOSBase                ;Export für amiga.lib
        XDEF    _stdout
        XDEF    _stdin

        move.l  #dosname,a1             ;Name der DOS-Lib
        moveq   #0,d0                   ;Version egal
        move.l  _SysBase,a6             ;Basis Exec
        jsr     _LVOOpenLibrary(a6)     ;DOS-Lib öffnen
        move.l  d0,_DOSBase             ;Zeiger notieren
        beq     exit                    ;falls Fehler

        move.l  d0,a6                   ;Base DOS-Lib
        move.l  #con,d1                 ;^Con-Definition
        move.l  #MODE_OLDFILE,d2        ;Console gibt es
        jsr     _LVOOpen(a6)            ;öffne sie
        tst.l   d0                      ;alles OK?
        beq     CloseDOS                ;wenn nicht
        move.l  d0,_stdout              ;Handle eintragen
        move.l  d0,_stdin

* 10 Zufallszahlen zwischen 1 und 999 ausgeben
* Anzeige formatiert als  "Zufallszahl nn = zzz"
* ---------------------------------------------
        moveq   #1,d3           ;Zähler auf 1
loop
        pea     999             ;Range vorgeben
        jsr     _RangeRand      ;Random-Funktion aufrufen
        addq.l  #4,sp           ;Stack korrigieren

        move.w  d0,-(sp)        ;Ergebnis zur Ausgabe
        move.w  d3,-(sp)        ;Zähler auch
        pea     fmt(pc)         ;Format-String
        jsr     _printf         ;nun alles drucken
        addq.l  #8,sp           ;Stack korrigieren

        addq    #1,d3           ;Zähler += 1
        cmp     #10,d3          ;fertig?
        ble.s   loop

* Einen Text ausgeben
* -------------------
        pea     msg             ;Format-String
        jsr     _printf
        addq.l  #4,sp           ;Stack zurück

* Auf Taste warten
* ----------------
        jsr     _getchar        ;warte auf Taste

* Consol-Window schließen
* -----------------------
        move.l  _DOSBase,a6
        move.l  _stdout,d1      ;Handle
        jsr     _LVOClose(a6)

* DOS-Lib schliessen
* ------------------
CloseDOS
        move.l  a6,a1                   ;DOS-Lib-Basis
        move.l  _SysBase,a6             ;Basis Exec
        jsr     _LVOCloseLibrary(a6)

exit
        rts

* Datenbereich
* ------------
fmt     dc.b    'Zufallszahl %2d = %3d',10,0
        cnop    0,4
msg     dc.b    10,'Tippen Sie Return ...',0
        cnop    0,4

_stdout  ds.l   1
_stdin   ds.l   1
_DOSBase ds.l   1

dosname  dc.b   'dos.library',0
         cnop   0,4
con      dc.b   'CON:0/12/640/200/Console',0
```

Bild 2. Die Umleitung der C-Ein-/Ausgabe über stdin und stdout in ein Konsol-Fenster

```
        opt     l+

        incdir  ":devpac/include/"
        include exec/exec_lib.i
        include exec/memory.i

        XREF.L  _printf
        XDEF    _main
_main

        moveq   #MEMF_PUBLIC,d1
        CALLEXEC  AvailMem      ;free mem -> d0
        move.l  d0,-(sp)
        pea     fmt             ;Format-String
        jsr     _printf
        addq.l  #8,sp           ;2 Longs vom Stack

        rts

fmt     dc.b    'Freier RAM = %ld Bytes',10,0
```

Bild 3. So einfach kann man die Größe des freien Speichers anzeigen

```
        opt     l+
* Demo _sprintf unter Intuition.
* Laufende Anzeige Free Mem in einem Window
* Programm abgewandelt aus Amiga Assembler-Buch M&T 90525
* (c) 1986/1987 P. Wollschlaeger

        incdir  ":devpac/include/"

        include exec/exec_lib.i
        include exec/memory.i
        include libraries/dos.i
        include libraries/dos_lib.i
        include intuition/intuition.i
        include intuition/intuition_lib.i
        include graphics/graphics_lib.i

GRAFIC  macro
        move.l  windowptr,a1            ;Adresse Window-Struktur
        move.l  wd_RPort(a1),a1         ;von da auf Rast Port
        CALLGRAF \1                     ;Grafik-Funktion
        endm

        XREF.L  _sprintf
        XDEF    _main
_main

* DOS-Library oeffnen
* -------------------
        lea     dosname,a1
        moveq   #0,d0
        CALLEXEC OpenLibrary
        tst.l   d0
        beq     abbruch
        move.l  d0,_DOSBase             ;Basis-Zeiger sichern

* Intuition Library oeffnen:
* --------------------------
        lea     intname,a1
        moveq   #0,d0
        CALLEXEC OpenLibrary
        tst.l   d0
        beq     closedos
        move.l  d0,_IntuitionBase       ;Basis-Zeiger sichern

* Graphics Library oeffnen
* ------------------------
        lea     grafname,a1
        moveq   #0,d0
        CALLEXEC OpenLibrary
        tst.l   d0
        beq     closeint
        move.l  d0,_GfxBase             ;Basis-Zeiger sichern

* Window oeffnen
* --------------
        lea     windowdef,a0            ;zeige auf Window-Struktur
        CALLINT OpenWindow              ;oeffne Window
        tst.l   d0                      ;ging was schief?
        beq     closegraf               ;wenn ja
        move.l  d0,windowptr            ;Window-Zeiger sichern
        moveq   #1,d0
        GRAFIC  SetAPen                 ;Zeichenfarbe weiß
```

Bild 4. Auch unter Intuition lassen sich die C-Funktionen sehr einfach einsetzen

```
* Hauptschleife
* -------------
loop    moveq   #MEMF_PUBLIC,d1      ;freien RAM
        CALLEXEC AvailMem             ;einlesen

        move.l  d0,-(sp)              ;Free Mem in d0
        pea     fmt                   ;Format-String
        pea     buffer                ;Ziel
        jsr     _sprintf
        lea     12(sp),sp             ;3 Longs vom Stack

        moveq   #60,d0                ;X-Position fuer Text
        moveq   #19,d1                ;Y
        GRAFIC  Move                  ;Move TO X,Y
        lea     buffer,a0             ;Text-Adresse
        moveq   #12,d0                ;Länge
        GRAFIC  Text                  ;Text zeichnen

        move.l  windowptr,a0          ;Vom unserem Window
        move.l  wd_UserPort(a0),a0    ;den Empfangsport
        CALLEXEC GetMsg               ;testen
        tst.l   d0                    ;Haben wir Post?
        bne     fini                  ;kann nur CLOSEWINDOW sein

        move.l  #25,d1                ;25/50 = 1/2 Sekunde
        CALLDOS Delay                 ;warten
        bra     loop                  ;und von vorn

fini    move.l  d0,a1                 ;Message ist in d0
        CALLEXEC ReplyMsg             ;antworten

closewindow
        move.l  windowptr,a0          ;Fenster zu
        CALLINT CloseWindow

closegraf
        move.l  _GfxBase,a1           ;Die Libs schließen
        CALLEXEC CloseLibrary

closeint
        move.l  _IntuitionBase,a1
        CALLEXEC CloseLibrary

closedos
        move.l  _DOSBase,a1
        CALLEXEC CloseLibrary

abbruch
        moveq   #0,d0                 ;keinen Fehler melden
        rts

W_Gadgets equ   WINDOWDRAG!WINDOWDEPTH!WINDOWCLOSE
W_Extras  equ   SMART_REFRESH!ACTIVATE

W_Title dc.b    ' Freier RAM z.Z. ',0

windowdef
        dc.w    200,20                ;links, oben
        dc.w    220,24                ;Breite, Hoehe
        dc.b    -1,-1                 ;Pens des Screen
        dc.l    CLOSEWINDOW           ;einizigess IDCMP Flag
        dc.l    W_Gadgets!W_Extras    ;Window Flags
        dc.l    0                     ;keine User-Gadgets
        dc.l    0                     ;keine User-Checkmark
        dc.l    W_Title               ;Titel des Window
        dc.l    0                     ;kein eigener Screen
        dc.l    0                     ;keine Super Bitmap
        dc.w    100,20                ;Min. Groesse
        dc.w    640,200               ;Max.
        dc.w    WBENCHSCREEN          ;Use Workbench Screen

intname         INTNAME               ;Namen der Libs aus Makros
grafname        GRAFNAME
dosname         DOSNAME
                cnop    0,4
fmt             dc.b    '%ld Bytes',0 ;Format-String
                cnop    0,4
buffer          ds.b    20
_IntuitionBase  ds.l    1
_GfxBase        ds.l    1
_DOSBase        ds.l    1
windowptr       ds.l    1
```

wie man ohne dieses Modul auskommen kann. Bei der Gelegenheit werden noch einige C-Funktionen vorgeführt und – darauf kommt es an – gezeigt, wie man die Ein-/Ausgabe der C-Funktionen auf ein eigenes Konsol-Fenster umleitet.

Die DOS-Library muß aus zwei Gründen geöffnet werden. Zuerst brauchen die C-Funktionen DOSBase für die Verwendung der DOS-Funktionen Read() und Write(), zum zweiten braucht man die Open-Funktion des DOS, um das Konsol-Fenster zu öffnen. Der Trick an der ganzen Geschichte ist nun, daß das Handle des Konsol-Fensters in die Variablen „_stdin" und „_stdout" (definiert in Amiga-Lib) eingetragen wird. Das ist die ganze Ein-/Ausgabe-Umleitung. Das Programm soll dann zehn Zufallszahlen in der Form „Zufallszahl n = zzz" formatiert ausgeben. Die Funktion RangeRand(x) erzeugt Pseudo-Zufallszahlen im Bereich 1...x, wobei x eine Konstante im Bereich von 1...65535 sein muß. Nur um die Eingabe zu demonstrieren, wird dann mit „_getchar" auf eine Taste gewartet. Deren Code steht übrigens danach in D0.B.

Bild 3 dient nur der Einstimmung auf das nächste Problem. Nach der schon bekannten Methode soll nun der freie Hauptspeicher angezeigt werden. Dank der Exec-Funktion AvailMem() ist das eine Kleinigkeit. Hier soll jedoch der freie Speicherplatz ständig in einem kleinen Fenster angezeigt werden, also auch, wenn andere Tasks laufen. Zusatzforderungen: Das Programm soll auch per Maus-Doppelklick von der Workbench aus gestartet werden können und das Fenster soll ein Close-Gadget haben, mit dem man das Programm beenden (und das Window verschwinden lassen) kann.

Damit bleibt nur eine Lösung: Das Programm muß unter Intuition laufen, printf() soll aber genutzt werden. Wie das geht, zeigt *Bild 4.*

Die Lösung funktioniert nach folgendem Prinzip: anstatt printf() wird sprintf() eingesetzt. Der einzige Unterschied: Die Ausgabe erfolgt in einen Puffer, dessen Adresse zusätzlich zu übergeben ist. Diesen Puffer kann man dann mit der Grafik-Funktion Text() ausgeben, wie hier geschehen, oder „IntuiText" einsetzen, wenn man noch interessante Texteffekte erzielen will.

Für Intuition-Kenner dürfte das Programm leicht verständlich sein, daher nur zwei Anmerkungen: Es sieht so aus, als ob das Programm nach der unfeinen Polling-Methode arbeitet. Das ist nicht der Fall. Durch den Aufruf der DOS-Funktion Delay() wird es für jeweils eine halbe Sekunde „suspendiert". Den Puffer für sprintf() sollten Sie immer etwas (hier acht Bytes) größer wählen als die Byte-Zählerei ergibt – es sei denn, Sie sind ein Fan von Guru-Meldungen. Bleibt abschließend festzustellen, daß ich zwar immer noch den dumpfen Verdacht habe, daß die C-Funktionen im ROM vergessener Debug-Code sind, aber sie sind offiziell dokumentiert und können deshalb mit Commodores Segen eingesetzt werden.

Literatur

[1] Commodore Amiga Inc: ROM Kernel Reference Manual Exec, Addison-Wesley, Reading, Massachusetts, 1986, ISBN 0-201-11099-7

[2] Commodore Amiga Inc: AmigaDOS-Handbuch, Verlag Markt & Technik, Haar bei München, 1988

[3] *Peter Wollschlaeger:* Amiga Assembler-Buch, Verlag Markt & Technik, Haar bei München, 1987

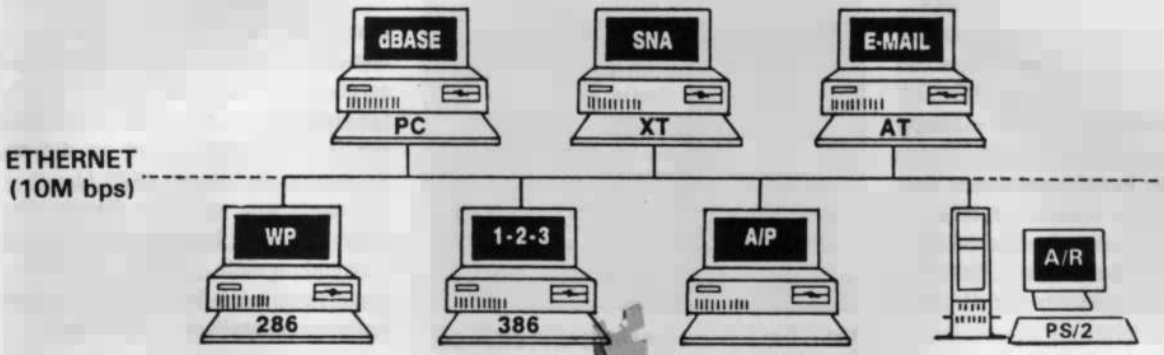
[4] *Peter Wollschlaeger:* Amiga-Programmierpraxis Intuition, Verlag Markt & Technik, 1988

D-Link™

Links Your PC's Together for Sharing of More Resources

The D-Link Ethernet and Twisted-pair adaptors provide the most cost-effective and easy-to-use local area networks for IBM PC/XT/AT and PS/2,™ as well as the IBM/NETBIOS and Novell's NetWare™ compatibles.

D-Link has the winning features: no dedicated file server, allows peripheral sharing, provides DOS 3.1 record and file locking, is transparent to DOS, provides diskless workstation capability, support electronic mail, on-line chatting, and screen monitor.



DATEX
SYSTEMS INC.
15-4 Fl., No.1, Fu-Hsing North Rd., Taipei, Taiwan, R.O.C.
Tel: 886-2-773-2980 Telex: 10971 DATEX
Fax: 886-2-781-5826

Distributor in W. GERMANY
PRODUTEC/SYSTEMS
Murrwiesinestraβe 18, D-7155 Oppenweiler
TEL: 07191-4001 FAX: 07191-4006

* IBM PC/XT/AT, NETBIOS and PS/2 are registered trademarks of International Business Machines Corp
* D-Link is a trademark of Datex Systems Inc
* Netware is a registered trademark of Novell Inc

Hardy Deutschlaender, Peter Horner

Omega-Net – Ein Peripherie-Netz für PCs

Teil 1: Konzeption und Hardware

Wer möchte nicht beliebige Geräte ohne Aufwand an seinen PC anschließen? Zusatzsteckkarten bieten gute Kompromisse, sind aber durch die Zahl der Steckplätze begrenzt. Anders bei Omega-Net: Zusätzliche Peripherie wird hier komplett extern installiert – im PC ändert sich nur die Software.

Durch die Vielseitigkeit des PCs sind immer neue Anwendungsgebiete für nahezu jedermann erschlossen worden – auch solche, die früher nur mit erheblichem finanziellem Aufwand an Hard- und Software realisierbar waren.

In der Industrie werden zur Kontrolle komplexer Fertigungsabläufe speicherprogrammierbare Steuerungen eingesetzt, beispielsweise das Siematic-S5-System der Siemens AG. Die Aufgabe solcher Systeme besteht – salopp gesprochen – hauptsächlich darin, Daten von Prozessen oder Fertigungsanlagen über digitale oder analoge Kanäle zu erfassen und Stellgrößen zu schalten. Nach festgelegten Algorithmen werden aus den Eingangszuständen die Sollwerte für die Ausgangszustände abgeleitet.

Mit einem derartigen System ist es zum Beispiel möglich, die Fensterrolläden und die Hausbeleuchtung während des Urlaubes automatisch und abwechslungsreich (also jeden Tag zu einer anderen Uhrzeit) zu steuern, eine Hausheizungs-Regelung oder eine Alarmanlage aufzubauen. Auch elektrische Modelleisenbahnen können mit einer speicherprogrammierbaren Steuerung als Ersatz für Relais- und TTL-„Gräber" bestens betrieben werden. Nur sind diese Steuerungen für den engagierten Amateur viel zu teuer, zumal ihre vielfältige Möglichkeiten – besonders bei relativ langsam ablaufenden Vorgängen – gar nicht genutzt werden.

Das Konzept

Eine preiswerte Alternative dazu bietet Omega-Net, von der Konzeption her ein „Token Passing"-Netz. Daran können 255 Knoten mit je 64 Ein- und Ausgangs-Kanälen, also insgesamt

16320 Eingangs- und
16320 Ausgangskanäle

angeschlossen und vom PC gesteuert werden. Die Knoten und Kanäle sind einzeln adressierbar. Das Netz wird mit Kommandos für folgende Funktionen gesteuert:

- Initialisieren eines Knotens,
- Konfiguration eines Knotens feststellen,
- Priorität eines Knotens festlegen,
- Schaltdauer ausgewählter Kanalgruppen begrenzen,
- ausgewählte Kanäle eines Knotens lesen und
- in ausgewählte Kanäle eines Knotens schreiben.

Die vom PC abgesetzten Kommandos werden im Netz bevorrechtigt behandelt.

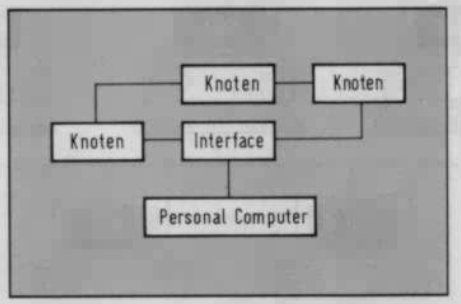


Bild 1. Das Schema des Omega-Netzes zur Übersicht

Die Ringleitung

Die Knoten sind über eine 2paarige Ringleitung (1 Paar als Kontroll-, 1 Paar als Datenleitung) miteinander verbunden. Die Ringleitung kann je nach Einsatz verdrillt, geschirmt oder ungeschirmt sein. Dadurch und in Verbindung mit den eingesetzten Differenz-Treiberstufen neuester Bauart werden Störungen durch Leitungseinstreuungen klein gehalten. Die Knoten werden von einem Prozessor 6802 mit spezieller Firmware im EPROM gesteuert, das auch die Knotenadresse enthält.

Die Steuerung des Netzes und die Verbindung mit dem PC erfolgt über einen speziellen Schnittstellen-Knoten, der nur einmal vorhanden ist und der für die notwendigen Impulse und Pegel-Anpassungen sorgt. Der PC selbst wird über seine V.24-Schnittstelle an diesen Knoten angeschlossen, die Peripherie wird an die anderen Knoten angeschlossen *(Bild 1)*.

Standardmäßig sind die Kanäle für TTL-Pegel ausgelegt. Da deren Leistung für viele Anwendungsfälle nicht ausreicht, sind Steckanschlüsse für Leistungsstufen und Entkoppelungsglieder vorgesehen.

Die Steuerung

Die Synchronisation der Knoten untereinander übernimmt auf der Kontroll-Leitung ein sogenanntes Token (daher der Name „Token Passing") und ein Host-Request. Die Netzwerk-Kommandos werden in Paketen (Frames) über die Datenleitung gesendet und empfangen.

Das Token ist ein Impuls, der von der Netzwerk-Steuerung eingesetzt wird und der die Sendeberechtigung für die einzelnen Knoten regelt. Es synchronisiert die Nutzung der Datenleitung, da ein Knoten nur dann auf die Datenleitung zugreift, wenn er das Token auf der Kontroll-Leitung identifiziert hat.

Mit dem Host-Request verschafft sich der PC bevorrechtigte Sendeberechtigung, d.h. jeder Knoten unterbricht seine augenblickliche Verarbeitung und erwartet die Netzwerk-Kommandos vom PC. Das Token wird aus dem Ring herausgenommen.

Zur Netzwerk-Steuerung gehört auch der „Token Release Request". Damit teilt der PC mit, daß eine Folge von Kommandos zu Ende ist. Das Token wird an der Stelle wieder in den Ring eingesetzt, an der es entfernt wurde.

Spezifische Daten

Die Verbindung zwischen dem PC und dem Netz stellt ein 7adriges Kabel mit einem 15poligen Stecker und einer 25poligen Buchse her *(Tabelle 1)*. Die Netzwerk-Knoten sind untereinander mit einem 2 x 2adrigen Kabel verbunden. Um Vertauschungen zu vermeiden, werden die Anschlüsse über Buchsen bzw. Stecker hergestellt.

Die Übertragungsgeschwindigkeit zwischen PC und Schnittstellen-Knoten sowie auf der Ringleitung beträgt 9600 Baud. Durch die Differenztreiber besteht zwischen den Netzwerk-Knoten keine direkte Masseverbindung. Zusätzlich kann eine galvanische Trennung der Knoten durch Ersatz der Empfängerstufen durch Optokoppler erreicht werden. Zu diesem Zweck sind die entsprechenden ICs gesockelt.

Die Impulsdauer der Synchronisationssignale wurde auf 4 µs für das Token und auf 8 µs für den Host-Request festgelegt.

Außer der Hardware gehört zu Omega-Net auch ein Programmpaket, das sämtliche Basis-Routinen für den Netz-Betrieb zur Verfügung stellt, insbesondere die Protokoll-Treiber für die Adressierung der Kanäle und die Datenausgabe sowie einen Handler für den Interrupt 0Ch. In bezug auf die Verarbeitung der Eingangssignale ist Omega-Net ein Echtzeitsystem.

Organisiert ist die Software in einer etwa 8 KByte großen Objekt-Bibliothek aus mehr als 20 Assembler-Routinen und Makros. Der Aufruf der Unterprogramme ist als CALL-Schnittstelle (Microsoft-Modell der Parameter-Übergabe) realisiert. Die Unterprogramme werden in die jeweiligen Anwendungsprogramme eingebunden. Ein bereits ausgeführtes Anwendungsprogramm, nämlich das Testprogramm für alle Grundfunktionen des Netzes, gehört als ausführbare Datei zum Paket.

Tabelle 1:
Netzwerk-Anschluß und Ringleitung

Pin	PC	Knoten	
1	Schirm1	1	Schirm2 *)
2	TxDat	2	RxDat
3	RxDat	3	TxDat
4	RTS	4	RTS
5	CTS	5	CTS
6	DSR	6	DSR
7	Gnd	7	Gnd
8	**)	8	DSR
20	DTR	15	DTR
Pin	**J1**	**Pin**	**P1**
1	Schirm1	1	Schirm2 *)
2	CTL+	2	CTL+
3	CTL-	3	CTL-
4	DAT+	4	DAT+
5	DAT-	5	DAT-
6	NC	6	NC
...	NC	...	NC
9	NC	9	NC

*) falls vorhanden
**) Brücke zwischen Pin 6 und 8 am PC-Stecker

Die Knoten

Die Knoten des Omega-Net sind die sogenannten Adreß-Decode-Module (ADM). Mit ihrem eigenen Mikroprozessor und EPROM und bilden sie die Schnittstelle zwischen Netz und angeschlossener Peripherie (*Bild 2*). Vom Prozessor gesteuert wird dabei

- die serielle Schnittstelle (UART-Simulation),
- das Netzwerk-Protokoll,
- die Interruptsteuerung bei einer Verarbeitungsanforderung vom PC und
- die Auswahl der Ein- und Ausgangskanäle.

Die im Omega-Net definierten Funktionen werden von der Firmware des ADM in Steuerbefehle umgesetzt. In der Firmware sind die drei Funktionsblöcke

- zyklische Abfrage der Eingangskanäle und Meldung an den PC,
- Empfangen, Decodieren und Ausführen der Steuerbefehle vom PC sowie
- Schreiben der Daten in die Ausgangskanäle

realisiert. Das Programm führt in seiner Hauptschleife drei Schritte durch: Zunächst wird der Zustand der Eingangskanäle gelesen. Ergibt sich dabei eine Änderung gegenüber dem letzten Lesen, so wird überprüft, ob es sich um eine gültige Zustandsänderung oder um das Prellen eines Einganges handelt. Eine gültige Änderung wird an den PC gemeldet; ist dagegen Prellen feststellbar, wird ein weiterer Schleifendurchlauf abgewartet. Unabhängig davon, ob Änderungen vorliegen oder nicht, werden im zweiten Schritt anstehende Kanaldaten ausgegeben. Als letztes wird der Watch-Dog-Timer zurückgesetzt.

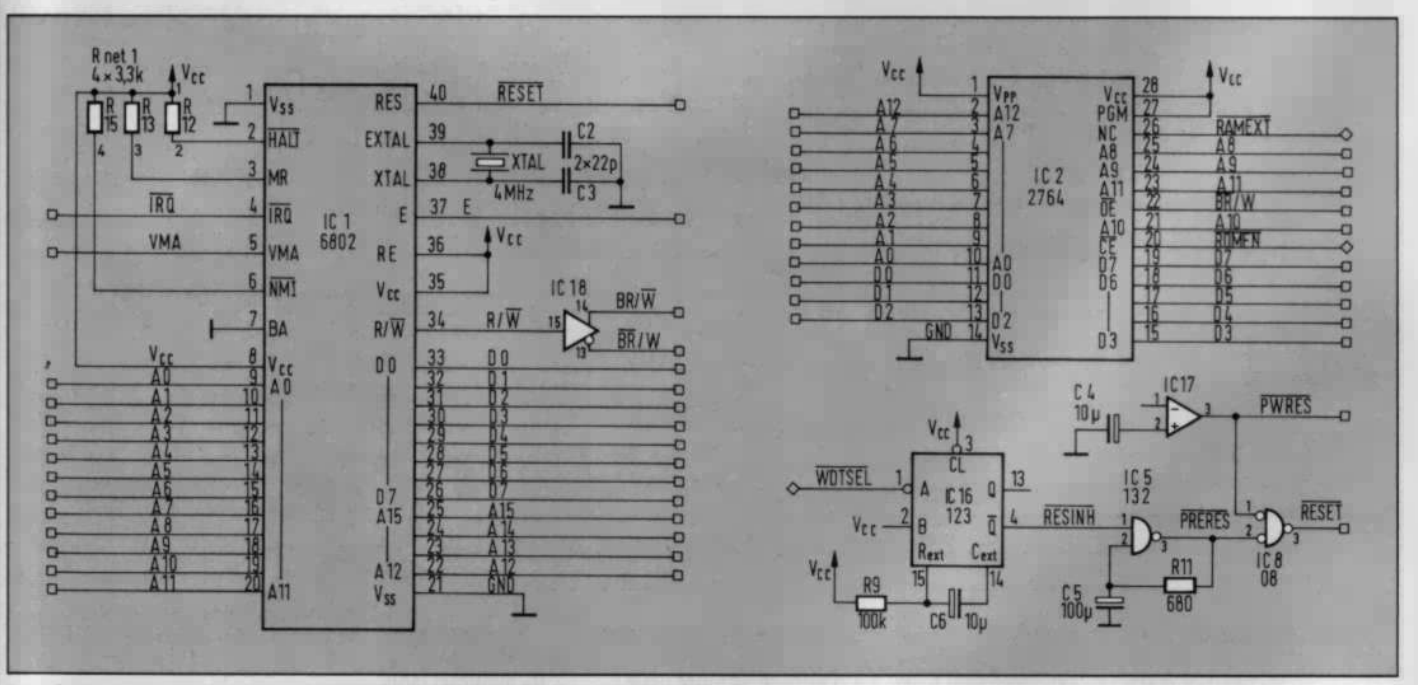


Bild 2. Der erste Teil der Schaltung: CPU, EPROM und Reset-Logik. Alle ICs sind 74LS-Typen außer IC 1, 2, 17 und 18

Die Hauptschleife läßt sich vom PC unterbrechen. Dies geschieht immer dann, wenn ein Knoten eine Netz-Funktion ausführen soll. Die Interrupt-Routine des Knotens kommuniziert mit dem PC über das Netzwerk-Protokoll. Sie sorgt gleichzeitig dafür, daß die Hardware-Anzeige (LEDs) und die zur Netzwerk-Kontrolle benötigten Steuersignale gesetzt werden. Sie decodiert den empfangenen Funktionscode, setzt ihn in Steuerbefehle um und führt diese aus.

Ausgaben auf die Kanäle werden in einem separaten Funktionsblock vorgenommen, den sowohl Hauptschleife als auch Interrupt-Routine aufrufen. Die Kanäle werden in Blöcken zu je 8 ausgegeben, und zwar wahlweise sofort oder im Abstand von 300 ms pro Block. Die Verzögerung kann immer dann benutzt werden, wenn die Stromversorgung der externen Verbraucher vor Überlastung geschützt werden soll.

Für die Token-Verarbeitung, den Watch-Dog-Timer und die Einschalt-Reset-Funktion ist zusätzliche Hardware vorhanden. Die Peripherie wird über 16 Ein- und 16 Ausgangs-Kanäle mit TTL-Pegel angeschlossen. Die Eingangs-Kanäle werden vom Prozessor zyklisch abgefragt und entprellt. Die +5 Volt Versorgungsspannung für einen Knoten muß TTL-Anforderungen genügen. Die Stromaufnahme beträgt maximal 600 mA.

Das Kernstück des Knotens ist der Prozessor 6802 (IC 1) mit 128 Byte RAM und ein 8-KByte-EPROM vom Typ 2764 (IC 2). Vom Daten-, Adreß- und Kontrollbus sind die Leitungen D0, A0, A1, A2 und R/W über Treiber geführt. Wegen der geringen Belastung werden alle anderen Signale nicht gepuffert. Die Taktfrequenz beträgt 4 MHz.

Beim Anlegen der Versorgungsspannung V_{cc} wird über IC 5 das Signal $\overline{PWRES}$ erzeugt. Damit werden die Ausgangskanäle auf logisch Null gesetzt. $\overline{PWRES}$ wird über IC 8 zu $\overline{RESET}$, das seinerseits über den Eingang RES des IC 1 ein Rücksetzen der CPU bewirkt. Gleichzeitig werden alle Flip-Flops und Schieberegister in ihren definierten Anfangszustand gesetzt.

Zur Überwachung der CPU ist ein Watch-Dog-Timer installiert. Dieser besteht aus den Monoflops IC 16 und 17 und muß vom Programm innerhalb von 300 ms durch das Signal $\overline{WDTSEL}$ angesprochen werden. Bleibt das Signal aus, so wird $\overline{RESET}$ erzeugt.

Die Ein-/Ausgabe-Adressen sind in den 64 KByte großen Adreßraum des 6802 entsprechend *Tabelle 2* abgebildet (memory mapped).

Tabelle 2: Ein-/Ausgabe-Adressen

Adresse	Bereich
0000h...1FFFh	RAM-Bereich
2000h...3FFFh	Ausgangskanäle
4000h...5FFFh	serieller Eingang
6000h...7FFFh	Eingangskanäle
8000h...9FFFh	serieller Ausgang
A000h...BFFFh	reserviert für externen Monitor
C000h...DFFFh	nicht belegt
E000h...FFFFh	ROM-Bereich

Adresse	Signal
2000h...2007h	OUT0
2008h...200Fh	OUT1
4000h	SERIN
6000h	INPORT
8010h	SEROUT
C000h	LOADEN
C001h	TOKHLD
C002h	CLRTIM
C003h	WDTSEL

Die Adreß-Decodierung *(Bild 3)* erfolgt in zwei Stufen in den ICs 4 und 22. IC 4 übernimmt die Adreß-Auswahl in 8 Blöcke

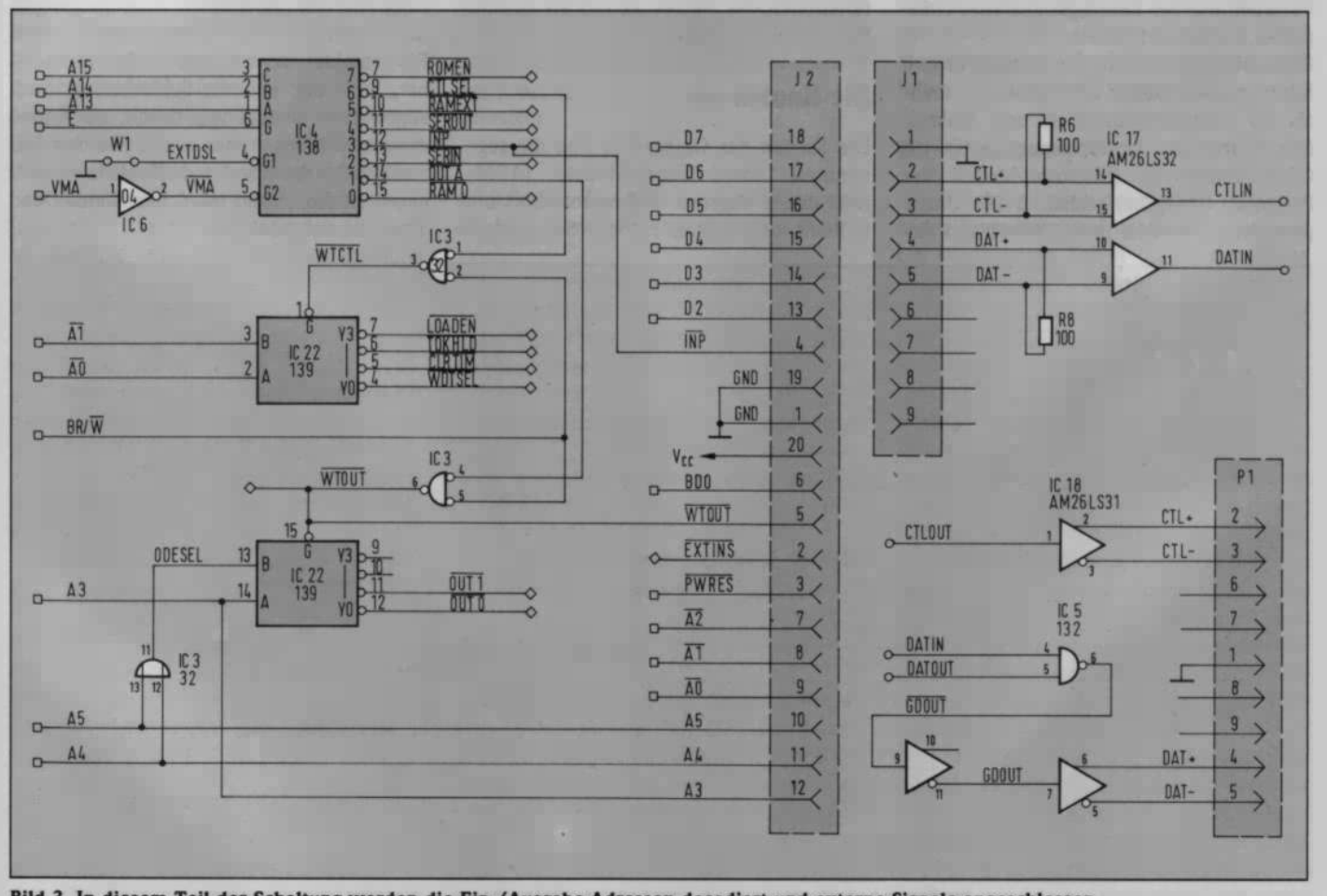
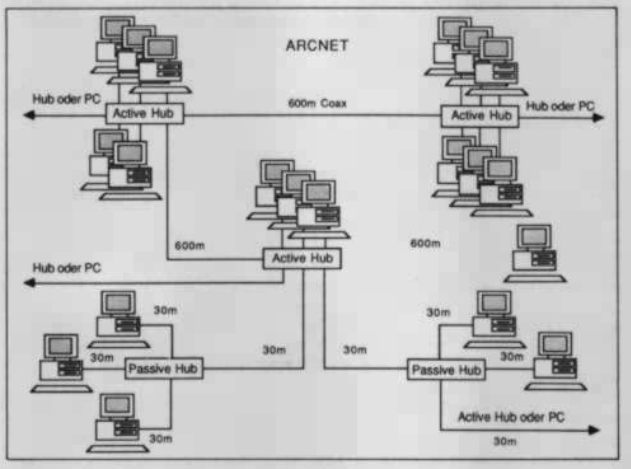


Bild 3. In diesem Teil der Schaltung werden die Ein-/Ausgabe-Adressen decodiert und externe Signale angeschlossen

Sie hätten gerne ein LAN

Sie besitzen Hochleistungs-PCs mit Bustaktraten bis 20 MHz? Sie planen, sowohl PC/XTs als auch ATs, als auch PS/2-Modelle mit Microkanal ins Netz zu integrieren? Sie möchten mit Novell NetWare arbeiten? Sie legen Wert auf Kompatibilität zum Arcnet-Standard? Sie nutzen Erweiterungen, wie etwa EGA-Grafik, EMS-Speicher und 3278/79-Emulationen? Sie wollen auch PCs ohne eigene Diskettenlaufwerke ins Netz einbinden? Sie sehen nicht ein, warum Sie einen langen Steckplatz für den Netzwerk-Adapter opfern sollen?

Und: Sie können sich Ausfallzeiten eines Netzwerkes nicht leisten?



Arcnet TCC!

Nur bei Adcomp!

Arcnet TCC Adapter: modernste Technik. Für höchste Ansprüche. Kompromißlos.

Weitere TCC-Features:

- belegt keine I/O-Adresse
- 2 KByte RAM-Puffer
- zwischen Standard-Arcnet und erweitertem Arcnet umschaltbar
- benötigt nur 16 KByte des System-Adressraums
- extrem lang und hart getestet
- brandneu: Safety-Net-Option – doppelte Performance im Normalfall, doppelte Sicherheit auf jeden Fall

Fragen Sie auch nach unseren weiteren Arcnet-Boards:

- Arcnet RX von Novell
- Arcnet SMC
- Arcnet Codenoll (optisches Arcnet)

Arcnet-Netze sind über aktive und passive Sternverteiler und flexibles Coax-Kabel sehr einfach zu installieren. Die Übertragungsrate beträgt 2,5 MBit/s. Das Protokoll (Token Passing) arbeitet ohne Kollisionen, wodurch die Übertragungsrate voll ausgeschöpft wird. Die maximale Entfernung beträgt in Arcnet-Netzen etwa 7 km. Arcnet-Boards unterschiedlicher Hersteller sind durch einen einheitlichen Arcnet-Communication-Controller-Chip untereinander kompatibel.

adcomp

ADVANCED COMMUNICATION PRODUCTS

Adcomp Datensysteme GmbH
Hauptgeschäftsstelle München
Olgastraße 15
8000 München 19
Telefon 089/12 00 50-0
Telex 5 216 271 adc
Telefax 080/12 00 50-45

Büro Hamburg
Kunhardtstraße 1
2000 Hamburg 20
Telefon 040/46 31 41
Telefax 040/46 31 69
Teletex 40 20 43 adc hmb

Büro Mönchengladbach
Alter Markt 24-25
4050 Mönchengladbach 1
Telefon 0 21 61/3 11 27
Telefax 0 21 61/3 22 17

Büro Neu-Isenburg
Bahnhofstraße 103
6078 Neu-Isenburg
Telefon 0 61 02/28 55
Telefax 0 61 02/2 50 23

Büro Stuttgart
Zettachring 10 a
7000 Stuttgart 80
Telefon 07 11/728 72 20
Telefax 07 11/728 71 50
Teletex 7 111 673 adc stgt

Tabelle 3: Buchsenleisten

Pin	J2	J3	J4
1	Gnd	Vcc	Vcc
2	EXTINS-	NC	IDAT12+
3	PWRES-	ODAT15+	IDAT13+
4	INP-	ODAT14+	IDAT14+
5	WTOUT-	ODAT13+	IDAT15+
6	BD0+	ODAT12+	IDAT11+
7	A2–	ODAT11+	IDAT10+
8	A1–	ODAT10+	IDAT09+
9	A0–	ODAT09+	IDAT08+
10	A5+	ODAT08+	NC
11	A4+	Gnd	NC
12	A3+	NC	Gnd
13	D2+	ODAT07+	IDAT04+
14	D3+	ODAT06+	IDAT05+
15	D4+	ODAT05+	IDAT06+
16	D5+	ODAT04+	IDAT07+
17	D6+	ODAT03+	IDAT03+
18	D7+	ODAT02+	IDAT02+
19	Gnd	ODAT01+	IDAT01+
20	Vcc	ODAT00+	IDAT00+

zu je 8 KByte (siehe Tabelle 2). IC 22 beinhaltet zwei voneinander unabhängige 4-Bit-Decodierer. Mit dem einen werden Knoten-interne Kontrollsignale gebildet; mit dem anderen wird über die Signale $\overline{OUT0}$ und $\overline{OUT1}$ sowie $\overline{A0}$, $\overline{A1}$ und $\overline{A2}$ einer der je 8 Ausgangskanälen (Kanäle 0...15) von IC 21 oder 23 selektiert (Adresse 2000h...200Fh). Die gesamte Zahl von 64 Kanälen pro Knoten ist nur mit einem Zusatzmodul erreichbar, das in der Fortsetzung dieses Beitrags beschrieben wird. Dieses Modul besitzt einen eigenen Decoder (IC E1) für die Adressen 2000h...203Fh. Da sich bei den beiden Decodern die Adressen 2000h...200Fh überschneiden, werden die unteren 16 Adressen mit IC 3 ausgeblendet.

Im Normalbetrieb liegt der Enable-Eingang des IC 4 an Masse, wodurch die Adreßauswahl freigegeben ist. Zu Servicezwecken wird der Jumper W1 entfernt und die Adreßauswahl von außen vorgenommen. Über die 20polige Buchsenleiste J2 *(Tabelle 3)* schließt man das Erweiterungsmodul an den Knoten an.

Jeder Knoten ist über einen 9poligen Sub-D Stecker (P1) und eine 9polige Sub-D Buchse (J1) in das Netz eingeschleift. Um auf der Ringleitung eine störungsarme Übertragung zu gewährleisten, wurden zur Ankoppelung Differenztreiber und -empfänger eingesetzt. Die gewählten Typen AM 26LS32 und AM 26LS31 (IC 17 und 18) kommen mit einer Versorgungsspannung von +5 Volt aus. Damit wird die sonst bei Operationsverstärkern nötige negative Spannung vermieden.

Die Empfänger erzeugen aus den Differenzsignalen CTL+, CTL–, DAT+ und DAT– die TTL-Signale CTLIN und DATIN. Die Ausgangssignale CTLOUT und DATOUT werden in den Treibern in die Differenzsignale CTL+ und CTL– bzw. DAT+ und DAT– umgesetzt.

Das Signal CTLIN liegt an dem mit 1 MHz getakteten Schieberegister IC 13 *(Bild 4)*. Dessen Parallelausgänge decodiert IC 14 jeweils 7 Taktzyklen nach einer Startflanke. Lowpegel am Adreßeingang B von IC 14 erzeugt am Ausgang Q4 das Signal $\overline{TOKEN}$. Das von den beiden softwaregesteuerten Signalen $\overline{TOKHLD}$ und BD0 verriegelbare Gatter des IC 5 setzt über $\overline{TOKLOD}$ das Flip-Flop IC 7. Damit ist gewährleistet, daß der Parallel-Seriell-Wandler IC 15 für die Dauer eines Taktes die an seinen Eingängen A...H liegende Information übernimmt. Die Eingänge E...H sind mit der Versorgungsspannung verbunden. Wenn das Signal IRQDET an den Eingängen A...D „low" ist, erzeugt IC 15 am Ausgang CTLOUT ein Token. CTLOUT ist über IC 6 und 8 auf den Preset-Eingang des IC 7 zurückgekoppelt und bewirkt, daß $\overline{SRLOAD}$ nur für einen Takt aktiv ist, was das Ende des Lade-Zyklus für IC 15 bedeutet.

Ein Knoten kann nur Daten senden, wenn er das Token festhält. Dazu wird per Software über BD0 und $\overline{TOKHLD}$ IC 12 gesetzt. Trifft ein Token ein, so läßt sich dies über IC 5 ($\overline{SETTOK}$) und IC 10/1 (TOKDET) vom Programm abfragen. $\overline{TOKLOD}$ bleibt inaktiv und verhindert so die oben beschriebene Token-Weitergabe. In diesem Status kann der Knoten nun Daten senden. Nachdem die Datenübertragung abgeschlossen ist, wird über BD0 und $\overline{TOKHLD}$ IC 12 und damit auch IC 10/1 gelöscht. Das ebenfalls software-gesteuerte Signal $\overline{LOADEN}$ gibt die oben beschriebene Erzeugung von CTLOUT frei. Der generierte Im-

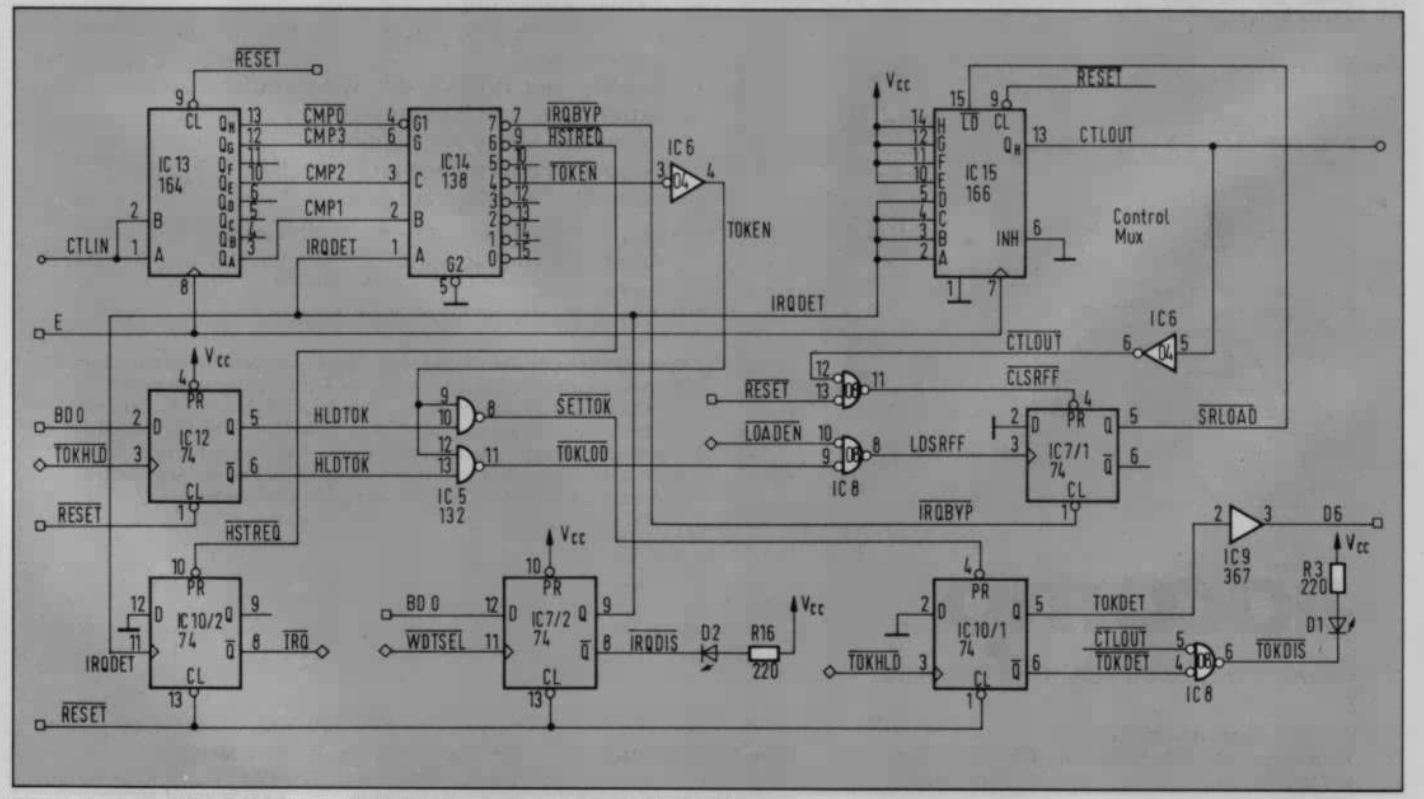


Bild 4. Der wohl komplexeste Teil der Schaltung: die Netzwerk-Kontrolle

puls ist wiederum ein Token, das an den nächsten Knoten weitergegeben wird.
Highpegel am Adreßeingang B des IC 14 erzeugt an dessen Ausgang Q6 das Signal HSTREQ, das über IC 10/2 die CPU mit einem maskierbaren Interrupt unterbricht. Bei der dann folgenden Interrupt-Verarbeitung werden IC 7/2 (und damit die rote Kontroll-LED D2) sowie IRQDET gesetzt. Nun liegen auch die Eingänge A...D des IC 15 auf Highpegel. Das Prozessor-gesteuerte Signal LOADEN gibt die bereits beschriebene Erzeugung von CTLOUT frei. Diesmal wird jedoch ein Host-Request-Impuls erzeugt. CTLOUT ist außerdem auf IC 10/2 zurückgekoppelt und löscht damit die Interrupt-Anforderung am 6802.
Folgt auf einen Host-Request ein zweiter, ohne daß der erste quittiert wurde, so darf ein Knoten diesen nicht als solchen erkennen. Dazu ist IRQDET auch an IC 14 gelegt. Anstelle von HSTREQ wird somit das Signal IRQBYP generiert. Dieses wiederum startet – unabhängig vom Signal LOADEN – die Erzeugung von CTLOUT. Dabei wird ein Host-Request-Impuls weitergegeben.

Die Kontroll-LED D2 ist eingeschaltet, wenn der Knoten einen Host-Request bearbeitet. Am Ende der Bearbeitung wird sie wieder ausgeschaltet.
Die Kontroll-LED D1 (gelb) zeigt die Position des Tokens im Ring an. Durch die kurzen Laufzeiten des Tokens glimmen die gelben LED aller Knoten. Die Intensität ist von der Anzahl der Knoten im Ring abhängig. D1 verlischt, wenn der Knoten einen Host-Request bearbeitet.
Die serielle Schnittstelle *(Bild 5)* besteht aus einem Treiber des IC 9 und einem Flip-Flop des IC 12. In Verbindung mit der Firmware simuliert die Schaltung den sonst notwendigen UART-Baustein. An dem ebenfalls über IC 9 geführten Signal EXTINS läßt sich erkennen, ob ein Erweiterungsmodul vorhanden ist.
Das Monoflop IC 16 wird über das Signal WTOUT vom Prozessor gesetzt. Die maximale Laufzeit beträgt rund 300 ms bzw. weniger, wenn es durch das Signal CLRTIM gelöscht wurde. Das Programm kann das Signal OUTSTB am Ausgang des Monoflops abfragen.

In die ICs 21 und 23 werden die Ausgangsdaten geladen. OUT0 und OUT1 selektieren eine Gruppe von je 8 Ausgangskanälen, von denen dazu einer durch A0, A1 und A2 adressiert wird. Die Daten werden bitseriell übertragen. Über eine 20polige Buchsenleiste (J3, Tabelle 3) sind die Ausgänge herausgeführt.
Die Eingangsleitungen werden ebenfalls über eine 20polige Buchsenleiste (J4, Tabelle 3) hereingeführt und liegen über Pull-Up-Widerstände an V_{cc}. Die ICs 19 und 20 multiplexen die Eingangsdaten auf die Leitungen D0 und D1 des Datenbusses. Alle Multiplexer werden von INP gleichzeitig selektiert; auch hier wird der Kanal durch die Adreß-Leitungen A0, A1 und A2 adressiert.
Tabelle 4 zeigt die Stückliste eines kompletten Netzwerk-Knotens.

Software

Die Omega-Net-Software ist das Bindeglied zwischen den Anwenderprogrammen und dem Netz. Sie ist im 8088/8086-Assembler

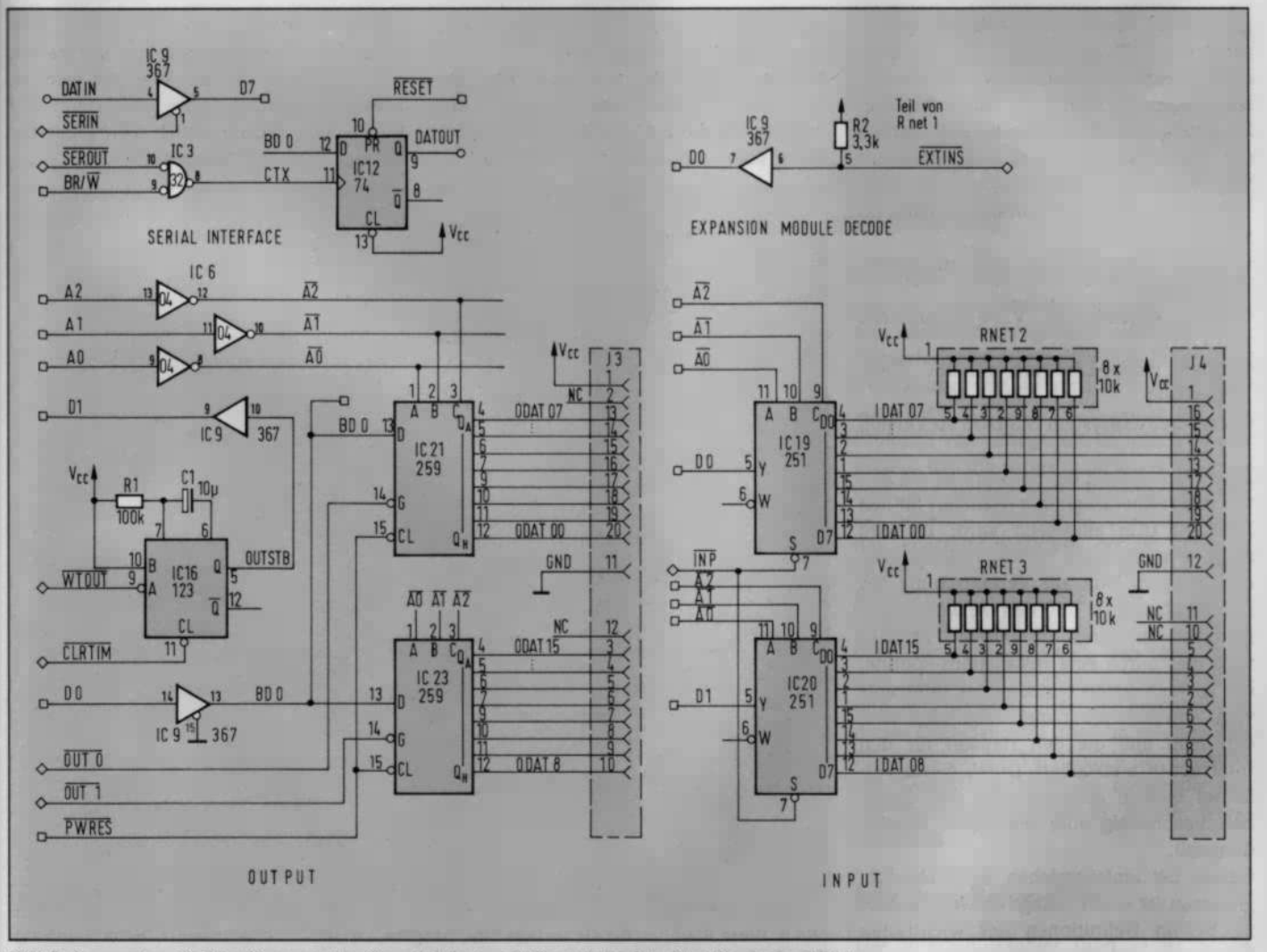


Bild 5. So werden die Ein-/Ausgangskanäle selektiert und die serielle Schnittstelle nachgebildet

Tabelle 4: ADM-Stückliste

Stück	Bezeichnung	Beschreibung
1	IC 1	6802
1	IC 2	2764
1	IC 3	74LS32
2	IC 4,14	74LS138
1	IC 5	74LS132
1	IC 6	74LS04
3	IC 7,10,12	74LS74
1	IC 8	74LS08
1	IC 9	74LS367
1	IC 13	74LS164
1	IC 15	74LS166
1	IC 16	74LS123
1	IC 17	AM26LS32
1	IC 18	AM26LS31
2	IC 19,20	74LS251
1	IC 22	74LS139
2	IC 21,23	74LS259
1	XTAL	Quarz 4 MHz
1	D 1	LED gelb, 5 mm
1	D 2	LED rot, 5 mm
3	C 1, 4, 6	Elko 10 µF, 16 V, stehend
2	C 2, 3	22 pF keramisch
1	C 5	Elko 100 µF, 16 V, stehend

Stück	Bezeichnung	Beschreibung
1	CB1	Tantal 10 µF, 16 V, stehend
14	C B2...15	100 nF RM 5 mm
1	Rnet 1 = R 2, 12, 13, 15	SIL 5 x 3,3 kΩ
2	R 3, 16	220 Ω
2	R 6, 8	100 Ω
2	R 1, 9	100 kΩ
1	R 11	680 Ω
2	Rnet 2, 3	SIL 8 x 10 kΩ alle Widerstände ¼ Watt, 5%
1	J 1	Steckverbinder D-Sub 9polig Buchse gewinkelt
1	J 2	Buchsenleiste 20polig gewink.
2	J 3, 4	Buchsenleiste 20polig stehend
1	J 5	Klemme für gedruckte Schaltung, 3polig
1	P 1	Steckverbinder D-Sub 9polig Stecker gewinkelt
1	W 1	Pfostenleiste 2polig mit Kurzschlußstecker
1	zu IC 1	IC-Sockel DIL 40
1	zu IC 2	IC-Sockel DIL 28
1	zu IC 17	IC-Sockel DIL 16

geschrieben und als Bibliothek (LIB) organisiert. Die Software benutzt nur Standardschnittstellen zum Betriebssystem wie den BIOS-Interrupt 0Ch oder ähnliche. Zur Steuerung der V.24-Schnittstelle werden die Ein-/Ausgabe-Adressen des Interface Controllers 8250 direkt angesprochen. Die Routinen laufen auf jedem kompatiblen PC mit

- einem Hauptspeicher von mindestens 128 KByte
- einer seriellen (= V.24-)Schnittstelle (COM1)
- einem Floppy-Laufwerk mit 360 KByte Kapazität
- dem Betriebssystem MS-DOS ab Version 2.11

Eine Besonderheit der Software ist die Art, wie das Treiberprogramm (Handler) für den Interrupt 0Ch eingesetzt wird: Es wird nicht, wie allgemein üblich, resident geladen, sondern läuft zur Vereinfachung des Datenaustausches mit den Anwenderprogrammen in deren Adreßraum. Erreicht wird dies durch eine Installations-Routine, die vor der ersten Nutzung des Interrupts vom Anwenderprogramm aufgerufen werden muß und die den Handler für den Interrupt 0Ch installiert. Durch dieses Verfahren bleiben außerdem die Basis-Routinen unabhängig vom jeweiligen Anwendungsfall.

Gerade bei umfangreichen Assembler-Programmen ist es unbedingt notwendig, sich zu Beginn Definitionen der wichtigsten Konstanten und Makros zu schaffen. In *Tabelle 5* sind die in von Omega-Net verwendeten Definitionen aufgeführt.

Mit diesen Definitionen ist es bereits möglich, die beiden Makros 'mpbyte' und 'mgbyte' für die Ein- und Ausgabe von Zeichen über die serielle Schnittstelle zu schreiben *(Bild 6)*. Dazu wird direkt in die Daten- und Status-Register des Controllers 8250 geschrieben oder von dort gelesen. Dies ist notwendig, da die benötigten Funktionen vom BIOS-Interrupt 14h nicht zur Verfügung gestellt werden.

```
Ein byte auf com(x) ausgeben

mpbyte MACRO     par1,par2,par3
       local     mpbyte_tx
       mov       cx,BIOS_com_wait    ; timer
       mov       dx,ls_reg           ; Line-Status-Register
       mov       ah,tcheck           ; Xmit.-Holding-Register
       call      cwait               ; .. muß leer sein
       jz        mpbyte_tx           ; time-out ?
       mov       par2,DOS_Err_TxReg  ; ja, zurück mit
       jmp       far ptr par3        ; .. Fehler
mpbyte_tx:
       mov       al,par1             ; nein, Ausgabe ins
       mov       dx,dt_reg           ; .. COMx-Data-Reg.
       out       dx,al
       ENDM

Ein byte von com(x) lesen

mgbyte MACRO     par1,par2,par3
       local     mgbyte_rx,mgbyte_return
       mov       cx,BIOS_com_wait    ; timer
       mov       dx,ls_reg           ; Line-Status-Reg.
       mov       ah,rcheck           ; warte, bis byte
       call      cwait               ; ... da ist
       jz        mgbyte_rx           ; time-out ?
       mov       par2,DOS_Err_TmOut  ; ja, zurück mit
       jmp       far ptr par3        ; .. Fehler
mgbyte_rx:
       mov       dx,dt_reg           ; nein, COMx-Data-
       in        al,dx               ; .. Register lesen
       mov       par1,al             ; Return Parameter
       and       ah,ovrrn            ; mehr als ein Byte?
       jz        mgbyte_return       ; nein, zurück
       mov       par2,DOS_Err_ovrrn  ; ja, zurück
       jmp       far ptr par3        ; .. mit Fehler
mgbyte_return:
       ENDM
```

Bild 6. Diese Routinen für die serielle Ein-/Ausgabe auf dem PC benutzen die Definitionen aus Tabelle 5

Tabelle 5: Standard-Definition und Makros

```
;BIOS-Equates
;Hardware Interrupts    EQU   0ch          ;Com1-Interrupt Vektor IRQ4
 BIOS_int_Com1          EQU   0bh          ;Com2-Interrupt Vektor IRQ3
 BIOS_int_Com2

;Software Interrupts
 BIOS_com               EQU   14h          ;RS232-Interrupt
 BIOS_com_WrChar        EQU   01h          ;ein Zeichen schreiben
 BIOS_com_RdChar        EQU   02h          ;ein Zeichen lesen
 BIOS_com_RdStat        EQU   03h          ;Status lesen
 BIOS_com_Wait          EQU   7ffffh       ;Warteschleifen Zähler

;BIOS-Daten Bereich
 BIOS_data_base         EQU   0040h        ;BIOS Daten Segment

;Adreß-Zeiger
 rs232_addr             DW    04h DUP(?)   ;Basis-Register Adressen
                                           ;für COM1 bis COM4

;Register des 8250 und 8259
 base                   EQU   3F0h         ;Basis
 dt_reg                 EQU   base + 08h   ;Daten-Register
 ie_reg                 EQU   base + 09h   ;Interrupt-Enable-Reg.
 ii_reg                 EQU   base + 0Ah   ;Interrupt-Ident.-Reg.
 lc_reg                 EQU   base + 0Bh   ;Line-Control-Register
 mc_reg                 EQU   base + 0Ch   ;Modem-Control-Register
 ls_reg                 EQU   base + 0Dh   ;Line-Status-Register
 ms_reg                 EQU   base + 0Eh   ;Modem-Status-Register

 intr_on                EQU   00001000b    ;MCR-Maske: 8250 Interrupts ein
 ie_slct                EQU   00001001b    ;IER-Bitmaske für MSI und RDRI
 im_reg                 EQU   21h          ;Interrupt-Mask-Register
 int_mask               EQU   0EFh         ;Maske, Bit 4 zu löschen

 rcheck                 EQU   01h          ;LSR-Maske: RX Daten verfügbar
 tcheck                 EQU   20h          ;LSR-Maske: Tx möglich
 ovrrn                  EQU   02h          ;LSR-Maske: Überlauf RX

 rls_int                EQU   06h          ;Line-Status-Interrupt
 rdr_int                EQU   04h          ;Receive-Data-Interrupt
 thre_int               EQU   02h          ;Xmit-Data-Interrupt
 ms_int                 EQU   00h          ;Modem-Status-Interrupt

 ocw2                   EQU   20h          ;Steuerwort des 8259
 eoi                    EQU   64h          ;Specific-end-of-Interrupt 4

 break_bit              EQU   40h          ;Maske für break
 dtr_on                 EQU   01h          ;Maske DTR Bit setzen
 rts_on                 EQU   02h          ;Maske CTS Bit setzen
 rsACK                  EQU   06h          ;Acknowledge Wert
 rsNAK                  EQU   15h          ;Not-Acknowledge Wert
 maxNAK                 EQU   0Ah          ;max. 10 NAKs erlaubt
 head_length            EQU   04h          ;Länge des Frame-Headers

;DOS-Equates
 DOS_fct                EQU   21h          ;DOS Funktionsaufruf
 DOS_fct_StIntV         EQU   25h          ;Setzen Interrupt Vektor
 DOS_fct_GtIntV         EQU   35h          ;Lesen Interrupt Vektor

;Fehler-Definitionen
 DOS_Err_Stmiss         EQU   20           ;DSR oder CTS fehlt
 DOS_Err_TxReg          EQU   19           ;Tx-Register voll
 DOS_Err_TmOut          EQU   27           ;Time Out
 DOS_Err_Ovrrn          EQU   25           ;Überlauf Rx-Register 8250
 DOS_Err_Chksm          EQU   28           ;Checksummen-Fehler
 DOS_Err_Garbg          EQU   29           ;verstümmelte Zeichen
 DOS_Err_Busy           EQU   30           ;ADM beschäftigt
```

```
;Macro-Definitionen

;Bereichsadresse in ein Register
 comm_adr       MACRO   reg,common
                push    ax
                mov     ax,seg common
                mov     reg,ax
                pop     ax
                ENDM

;Register auf den Stack
 push_regs      MACRO
                pushf
                push    es
                push    ds
                push    si
                push    di
                push    dx
                push    cx
                push    bx
                push    ax
                ENDM

;Register vom Stack
 pop_regs       MACRO
                pop     ax
                pop     bx
                pop     cx
                pop     dx
                pop     di
                pop     si
                pop     ds
                pop     es
                popf
                ENDM
```

An dieser Stelle muß festgelegt werden, wie die Datenübergabe zwischen Treiber- und Anwenderprogramm aussehen soll. Normalerweise würde sich eine Ringpuffer-Verwaltung, wie sie auch in anderen Software-Paketen realisiert ist, anbieten. Das Protokoll von Omega-Net ist jedoch so aufgebaut, daß die Ausgabe nicht zwischengespeichert werden muß: Eingabedaten von der seriellen Schnittstelle müssen wegen des Echtzeit-Charakters der Anwendung ohnehin direkt verarbeitet werden. Deshalb wird der Übergabebereich OMEGA$a ausschließlich statisch verwaltet. Er ist in drei Abschnitte aufgeteilt: den Statistikteil zum Zählen der Interrupts, den Ausgabe- und den Eingabepuffer. Die Beschreibung dieser Pufferbereiche wird im zweiten Teil folgen, da für ihr Verständnis die Kenntnis des Omega-Net-Protokolls notwendig ist. Der Statistikbereich ist in *Tabelle 6* beschrieben.

Interrupt-Behandlung

Der Interrupt-Handler (Bild 7) ist neben den Protokoll-Treibern das Herzstück der

Tabelle 6: Definition Übergabebereich OMEGA$a (Teil 1)

Segment-Struktur		
Name	Format	Beschreibung
rls_cnt	W	Anzahl Line-Status-Interrupts
rdr_cnt	W	Anzahl Receive-Interrupts
thr_cnt	W	Anzahl Transmitter-Hold.-Reg.-Int.
ms_cnt	W	Anzahl Modem-Status-Interrupts
rdr_sum	W	Summe aller Receive-Interrupts
nak_cnt	W	Anzahl Not-Acknowledges
nak_sum	W	Summe Not-Acknowledges
ADM_session	W	Summe aller ADM-Session
err_cnt	W	davon fehlerhafte

Software: Hier werden die asynchron ankommenden Datenpakete der Knoten registriert und an die Anwenderprogramme weitergeleitet. Dabei wird der BIOS-Interrupt 0Ch verwendet, also der im Controller 8259 vorgesehene, im BIOS aber nicht ausgenutzte Hardware-Interrupt (IRQ4) der seriellen Schnittstelle. Bei jedem auftretenden Ereignis (z. B. Änderung eines Schnittstellen-Signals oder Empfang eines Zeichens) wird der Handler aufgerufen und das Ereignis in den dafür vorgesehenen Feldern des Übergabebereichs OMEGA$a registriert. Durch entsprechende Programmierung des Interrupt-Enable-Registers im 8250 kann man die zulässigen Hardware-Interrupts beeinflussen. In der Version 1.00 werden zur Kontrolle der V.24-Schnittstelle der Receive-Data-Ready- und der Modem-Status-Interrupt verwendet.

Im Handler werden direkt nach dem Aufruf durch den Interrupt alle Register „gerettet" und das DS-Register auf die Adresse des Übergabebereichs gesetzt. Anhand des Interrupt-Identification-Registers läßt sich dann feststellen, welcher Interrupt vorliegt. Änderungen des Modem-Status werden lediglich registriert, während beim Empfang eines Zeichens sofort nach der Statistik die eigentliche Verarbeitung beginnt. Auch diese ist selbstverständlich vom Omega-Net-Protokoll abhängig und wird deshalb im zweiten Teil beschrieben.

Der Handler endet mit einem „end of specific interrupt", der das Control-Register des 8259 zurücksetzt, und mit dem obligatorischen IRET.

Mit der Definition eines Interrupt-Handlers allein ist es jedoch nicht getan. Es werden zusätzlich Routinen benötigt, die diesen in-

```
SUBTTL <Code Segment> HANDLER :  Interrupt 0Ch Treiber

        DB    '**HANDLE**'
dataseg     DW    ?
handler     PROC FAR  ; Einsprungadresse des INT 0Ch
        NOP           ; für DEBUG durch INT 3 ersetzen
        push_regs     ; der Unterbrochene dankt es uns
        ASSUME    DS:omega$a      ; OMEGA.net-Bereich
        comm_adr  DS,omega$a      ; lade DS-Register
        MOV  ibfmod,0105h         ; Verarbeitung im Handler

repoll:               ; auf ein neues
        MOV  DX,ii_reg ; das ist das Interrupt-Ident.-
        IN   AL,DX     ; .. Reg.
        CMP  AL,01h    ; Interrupts vorhanden ?
        JE   int_end   ; Nein, Ende der Fahnenstange
        MOV  AH,AL     ; Ja, rette Inhalt von IIR

        MOV  AL,AH     ; hole IIR
        AND  AL,rdr_int
        CMP  AL,rdr_int; ist es der, den wir suchen?
        JNE  recv_line ; .. leider nein

        MOV  DX,ls_reg ; schaut her, er ist's! Im Line-
        IN   AL,DX     ; .. Status-Reg. steht, ob das Byte
        AND  AL,rcheck ; .. schon da ist
        JZ   repoll    ; doch nicht, versuch's nochmal

        MOV  DX,dt_reg ; hurra, wir haben ein Byte, lesen
        IN   AL,DX     ; .. es, erhöhen den Rx-Interrupt-
        INC  rdr_cnt   ; .. Zähler
        INC  rdr_sum   ; .. und machen etwas Statistik
        MOV  BX,rdr_cnt; benutze den Zähler als Index
        MOV  ibfADM</BX-1/>,AL ; speichere das Zeichen
        CMP  BL,head_length ; war es der 4.Interrupt?
        JNE  int_end   ; .. nein, dann war's das für jetzt
        JMP  SHORT host_req_check ; ja, jetzt wird's ernst

recv_line:            ; kein rdr-int
        MOV  AL,AH     ; hole IIR
        AND  AL,rls_int; war es vielleicht
        CMP  AL,rls_int; .. ein Receiver-Line-Status-Int?
        JNE  thr_empty ; auch nicht, weiter versuchen
        MOV  DX,ls_reg ; lies das Line-Status-Reg
        IN   AL,DX
        INC  rls_cnt   ; .. und zähle mit
        JMP  repoll    ; versuch's nochmal

thr_empty:            ; kein rls-int
        MOV  AL,AH     ; hole IIR
        AND  AL,thre_int ; war es ein Transmitter-Holding-
        CMP  AL,thre_int ; .. Register-Empty-Interrupt?
        JNE  modem_stat; auch nicht, ein letztes Mal
        MOV  DX,ii_reg ; lies das IIR
        IN   AL,DX
        INC  thr_cnt   ; .. und zähle mit
        JMP  repoll    ; versuch's nochmal

modem_stat:           ; kein thre-int
        MOV  AL,AH     ; hole IIR
        AND  AL,ms_int ; war es vielleicht
        CMP  AL,ms_int ; .. ein Modem-Status-Int?
        JNE  int_end   ; auch nicht, dann ist es egal
        MOV  DX,ms_reg ; lies das Modem-Status-Reg
        IN   AL,DX
        INC  ms_cnt    ; .. und zähle mit
        JMP  repoll    ; versuch's nochmal

int_end:              ; hier geht's immer lang am Ende
        MOV  DX,ocw2   ; .. mit einem EOI an den 8259
        MOV  AL,eoi
        OUT  DX,AL
        pop_regs
        IRET           ; dies sollte man nicht vergessen

;----  nach 4 Byte header geht's jetzt richtig los ----
host_req_check:
        INC  ADM_session    ; eine weitere ADM-Session
        MOV  ibfmod,0104h   ; Verarbeitung Checksumme
        MOV  ibfcod,DOS_Err_Chksum    ; vorbesetzen Fehler
        MOV  AL,ibfADM ; errechne die Header-Checksumme
        ADD  AL,ibffct ; .. und vergleiche sie mit der vom
        ADD  AL,ibfnch ; .. ADM geschickten; bei Gleich-
        CMP  AL,ibfcsm ; .. heit antworten wir mit
        JE   ack_head  ; .. ACK
        JMP  FAR PTR error_count ; ungleich, Fehler zählen
ack_head:
        MOV  ibfmod,0103h   ; vorbesetzen Fehlermeldung
        mpbyte rsACK,ibfcod,reset ; sende ein ACK
        mgbyte AL,ibfcod,reset    ; wegen Ringleitung
        MOV  ibfcod,DOS_Err_Garbg ; mögl. Leitungsfehler
        CMP  AL,rsACK  ; .. gleich zurück kommt, wenn es
        JNE  error_count    ; ..ungleich ist, sofort RESET

        CALL omgrd     ; alles O.K., lies die Daten
        MOV  AX,ibfcod ; Fehler beim Lesen?
        AND  AX,AX
        JNZ  error_count    ; ja, dann Ende

        PUSH DS        ; bevor wir die Kontrolle abgeben,
        MOV  AX,CS:dataseg ; .. sichern wir lieber den
        MOV  DS,AX     ; .. Rückzug, damit der Nutzer sich
        CALL rswrk$    ; jetzt endlich austoben darf
        POP  DS        ; es hat sich also gelohnt
        MOV  ibfcod,0000h   ; Lösche Fehler-Anzeige
        MOV  rdr_cnt,0000h ; setze Interrupt-Zähler zurück
        JMP  int_end

error_count:
        INC  err_cnt   ; erhöhe ADM-Session-Fehlerzähler
        MOV  rdr_cnt,0000h  ; Lösche Fehler-Anzeige
        JMP  int_end
handler     ENDP
```

Bild 7. Erst durch die Programm-Kommentare wird dieser Interrupt Handler durchschaubar

```
SUBTTL <Code Segment> CINSTL :  Handler installieren

        DB    '**CINSTL**'
CInstl  PROC  FAR
        NOP                             ; für DEBUG durch INT3

        MOV   AH,DOS_fct_GtIntV         ; Inhalt des Vektors 0Ch
        MOV   AL,BIOS_Int_Coml          ; .. nach ES:BX holen
        INT   DOS_fct                   ; sollte 0:0 sein

        PUSH  DS                        ; neue Handler-Adresse
        LDS   DX,handler_adr            ; .. nach DS:DX
        MOV   AX,ES
        MOV   CX,DS
        CMP   CX,AX                     ; schon installiert?
        JNE   set                       ; .. Nein !
        CMP   DX,BX

        JE    SHORT cinstl_return       ; .. Ja, nicht nochmal
set:
        MOV   AH,DOS_fct_StIntV         ; neue Adresse in den
        MOV   AL,BIOS_Int_Coml          ; .. Vektor einhängen
        INT   DOS_fct                   ; .. DS:DX stehen schon
        POP   DS                        ; .. richtig für die
        PUSH  DS                        ; .. Funktion
        MOV   org_vec_seg,ES            ; Original-Vektor sichern
        MOV   org_vec_off,BX

        MOV   AX,DS                     ; Datensegment des Pro-
        MOV   CS:dataseg,AX             ; .. gramms in den Code
                                        ; .. des INT 0Ch übertrg.
        comm_adr ES,omega$a             ; Übergabebereich
        MOV   BX,OFFSET ibfcsm_data     ; .. löschen
next_clear:
        MOV   BYTE PTR ES:</BX/>,00h
        DEC   BX
        JNL   next clear

CInstl_return:
        POP   DS                        ; das war's dann
        RET
CInstl  ENDP


SUBTTL <Code Segment> CUNSTL :  Handler entfernen

        DB    '**CUNSTL**'
CUnstl  PROC FAR
        NOP             ; für DEBUG durch INT3 ersetzen
        CALL cdsabl     ; COM1-Interrupts stillegen
        MOV   AH,DOS_fct_GtIntV     ; Inhalt des Vektors 0Ch
        MOV   AL,BIOS_Int_Coml      ; .. nach ES:BX holen
        INT   DOS_fct

        PUSH DS
        LDS   DX,handler_adr        ; Handler-Adresse
        MOV   AX,ES                 ; ..nach DS:DX
        MOV   CX,DS
        CMP   CX,AX                 ; schon installiert ?
        JNE   cunstl_return         ; .. Nein!
        CMP   DX,BX                 ; .. das war's dann
        JNE   cunstl_return

        POP   DS                    ; Ja, jetzt wird
        PUSH DS                     ; .. entfernt
        MOV   DX,org_vec_off
        MOV   DS,org_vec_seg
        MOV   AH,DOS_fct_StIntV
        MOV   AL,BIOS_int_Coml
        INT   DOS_fct

CUnstl_return:
        POP   DS
        RET
CUnstl  ENDP
```

Bild 8. Die Installations-Routinen instalieren oder entfernen den Interrupt-Handler aus Bild 7

```
SUBTTL <Code Segment> CENABL :  Interrupts 8250 aktiv

        DB    '**CENABL**'
CEnabl  PROC FAR
        NOP             ; für DEBUG durch INT3 ersetzen
        CLI             ; Interrupts ausschalten (hier ja)

        MOV   DX,mc_reg ; Modem-Control-Register zurücks.
        IN    AL,DX
        MOV   DX,ls_reg ; Line-Status zurücksetzen
        IN    AL,DX
        MOV   DX,dt_reg ; Receive-Data zurücksetzen
        IN    AL,DX
        MOV   DX,ms_reg ; Modem-Status zurücksetzen
        IN    AL,DX

        IN    AL,im_reg ; Interrupt-Controller 8259
        AND   AL,int_mask
        OUT   im_reg,AL ; .. initialisieren

        MOV   DX,ie_reg ; Interrupt-Enable-Register des
        MOV   AL,ie_slct      ; .. 8250 einstellen
        OUT   DX,AL

        MOV   DX,mc_reg ; Modem-Control des 8250 einstell.
        IN    AL,DX
        OR    AL,intr_on ; ..Bit 3 gesetzt
        OUT   DX,AL

        comm_adr ES,OMEGA$a ; Übergabebereich
        MOV   ES:rdr_cnt,00h ; Rx-Interrupt Zähler zurück-
        STI                  ; .. setzen, INT erlauben
        RET
CEnabl  ENDP

SUBTTL <Code Segment> CDSABL :  Interrupts 8250 stilleg.

        DB    '**CDSABL**'
CDsabl  PROC FAR
        NOP             ; für DEBUG durch INT3 ersetzen
        CLI             ; Interrupts ausschalten

        MOV   DX,ie_reg ; Interrupt-Enable-Register des
        XOR   AL,AL     ; .. 8250 löschen
        OUT   DX,AL
        MOV   DX,im_reg ; IRQ4 des 8259 ausschalten
        IN    AL,DX
        OR    AL,NOT int_mask
        OUT   DX,AL
        STI             ; .. setzen, INT erlauben
        RET
CDsabl  ENDP
```

Bild 9. Erst wenn der Interrupt-Handler installiert ist, dürfen die Interrupts mit diesem Programm freigegeben werden

stallieren und – da man mit „verbogenen" Interrupt-Vektoren eine Menge Unheil anrichten kann – auch wieder entfernen. Außerdem braucht man Routinen, die den Handler aktivieren, wenn er gebraucht wird, und bei Bedarf auch wieder stillegen. Wozu dies gut ist, wird im zweiten Teil bei der Beschreibung der Protokoll-Treiber deutlich.

Beginnen wir mit dem Anfang, der Installation *(Bild 8)*. Der Interrupt-Handler wird im Hardware Interrupt 0Ch für die erste serielle Schnittstelle mit den Funktionen 25h und 35h des DOS-Interrupts 21h eingehängt oder daraus entfernt. Dieses Verfahren ist schon so oft beschrieben worden, daß es keiner näheren Erläuterung bedarf. Ist der Handler bereits installiert bzw. nicht vorhanden, so haben die Funktionen keine Wirkung. Soll der Interrupt-Handler entfernt werden, wenn er noch aktiv ist, so wird er erst stillgelegt.

Beim Entfernen werden keine CLI/STI-Befehle verwendet. Das ist kein Fehler, denn nach dem Stillegen sollte eigentlich kein Interrupt mehr über COM1 erscheinen.

Als nächster Schritt muß der bereits installierte Interrupt-Handler aktiviert bzw. stillgelegt werden *(Bild 9)*. Dazu werden das Interrupt-Enable-Register (Bits 3 und 0) und das Modem-Controll-Register (Bit 3) des Bausteins 8250 sowie das Interrupt-Mask-Register (Bit 15,14,...,1) des Bausteins 8259 verändert. Um den 8250 in einen definierten Ausgangszustand zu versetzen, werden alle seine Register einmal

```
SUBTTL <Code Segment> CWAIT :  Warten auf Status des 8250

        DB    '**CWAIT **'
cwait   PROC FAR
        NOP             ; für DEBUG durch INT3 ersetzen

        PUSH BX             ; temp. register retten
        PUSH AX             ; Zeichen in AL retten
        MOV  BL,04h         ; globaler Zähler

cwait_loop:
        IN   AL,DX          ; lies Status und rette ihn
        MOV  BH,AL          ; .. temporär in BH
        AND  AL,AH          ; maskiere die Bits aus, auf
        CMP  AL,AH          ; .. die wie warten und test?
        JZ   cwait_exit     ; alle an, zurück mit ZF ein
        LOOP cwait_loop     ; warte,warte nur ein Weilchen

        DEC  BL
        JG   cwait_loop
        OR   AH,AH          ; time out, lösche ZF
cwait_exit:
        POP  AX             ; restauriere alles temporär
        MOV  AH,BH          ; .. zwischengespeicherte
        POP  BX
        RET
cwait   ENDP
```

Bild 10. Die Warte-Schleife dieser Routine wird verlassen, wenn entweder das gewünschte Status-Flag des 8250 gefunden wird oder der Timer der Warterei ein (definiertes) Ende setzt

```
SUBTTL DTROFF -- DTR-Signal "0"

        DB    '**DTROFF**'
dtroff PROC FAR
        NOP             ; für DEBUG durch INT3 ersetzen
        MOV  DX,mc_reg  ; Lies Modem-Control-Register
        IN   AL,DX      ; .. lösche DTR-Bit und
        AND  AL,NOT dtr_on
        OUT  DX,AL      ; .. schreibe zurück
        RET
dtroff ENDP


SUBTTL DTRON -- DTR-Signal "1"

        DB    '**DTRON **'
dtron  PROC FAR
        NOP             ; für DEBUG durch INT3 ersetzen
        MOV  DX,mc_reg  ; Lies Modem-Control-Register
        IN   AL,DX      ; .. setze DTR-Bit und
        OR   AL,dtr_on
        OUT  DX,AL      ; .. schreibe zurück
        RET
dtron  ENDP
```

Bild 11. Mit diesen Routinen wird die Leitung DTR (Data Terminal Ready) der V.24-Schnittstelle geschaltet

```
SUBTTL RTSOFF -- RTS-Signal "0"

        DB    '**RTSOFF**'
rtsoff PROC FAR
        NOP             ; für DEBUG durch INT3 ersetzen
        MOV  DX,mc_reg  ; Lies Modem-Control-Register
        IN   AL,DX      ; .. lösche RTS-Bit und
        AND  AL,NOT rts_on
        OUT  DX,AL      ; .. schreibe zurück
        RET
rtsoff ENDP

SUBTTL RTSON -- RTS-Signal "1"

        DB    '**RTSON **'
rtson  PROC FAR
        NOP             ; für DEBUG durch INT3 ersetzen
        MOV  DX,mc_reg  ; Lies Modem-Control-Register
        IN   AL,DX      ; .. setze RTS-Bit und
        OR   AL,rts_on
        OUT  DX,AL      ; .. schreibe zurück
        RET
rtson  ENDP
```

Bild 12. Wie in Bild 11 geht es hier um das Schalten eines V.24-Ausganges am PC: Um das Signal RTS (Request To Send)

```
SUBTTL <Code Segment> DSRsta : Testen Status DSR Signal

        DB    '**DSRSTA**'
dsrsta PROC FAR
        NOP

        MOV  CX,BIOS_com_wait ; timer
        MOV  DX,ms_reg  ; das Bit, auf das wir warten, ist
        MOV  AH,20h     ; .. das Bit 5 vom Modem-Status-
        CALL cwait      ; .. Register
        MOV  AX,0000h   ; nimm an, es kam nicht
        JNZ  dsr_return ; time-out ?
        INC  AX         ; nein, dann mit .true. zurück
dsr_return:
        RET
dsrsta ENDP

SUBTTL <Code Segment> CTSSTA : Testen Status CTS Signal

        DB    '**CTSSTA**'
 ctssta          PROC FAR
        NOP

        MOV  CX,BIOS_com_wait    ; timer
        MOV  DX,ms_reg  ; das Bit, auf das wir warten, ist
        MOV  AH,10h     ; .. das Bit 4 vom Modem-Status-
        CALL cwait      ; .. Register
        MOV  AX,0000h   ; nimm an, es kam nicht
        JNZ  cts_return ; time-out ?
        INC  AX         ; nein, dann mit .true. zurück
 cts_return:
        RET
 ctssta          ENDP
```

Bild 13. Hier nun werden die Status-Eingänge der V.24-Schnittstelle vom PC abgefragt

gelesen. Das Interrupt-Flag ist natürlich während dieser Operationen gelöscht.

Weitere Routinen

In allen Lese- und Schreibroutinen müssen bestimmte Betriebszustände (z. B. 'Zeichen empfangen') getestet werden. Es wird dabei so lange gewartet, bis die getestete Bedingung eintritt oder eine Zeitschleife abgelaufen ist.

Diese Aufgabe übernimmt die Warte-Routine in *Bild 10*, die mit DX ein wählbares Status-Register des 8250 (normalerweise das Line-Status-Register LSR oder das Modem-Status-Register MSR) auf ein vorgegebenes Status-Bit hin überprüft. Dies geschieht in einer Zeit-Schleife über das BX- und CX-Register, deren Werte vor dem Aufruf festgelegt werden müssen. Nach Verlassen der Routine sind beide Werte wieder undefiniert. Ist nach Ablauf des Timers das gewünschte Status-Bit nicht gesetzt, so endet die Routine mit einer Fehlermeldung, d. h. das Zero-Flag wird gelöscht. Sonst wird die zu prüfende Bedingung mit gesetztem Zero-Flag als erfüllt gemeldet.

Um die serielle Schnittstelle optimal auszunutzen, sind noch weitere Routinen notwendig. Die Ausgangssignale DTR und RTS müssen gezielt ein- und ausgeschaltet werden, und der Status der Eingangssignale DSR und CTS muß geprüft werden können. Dies erledigen die Programme in *Bild 11...13.*

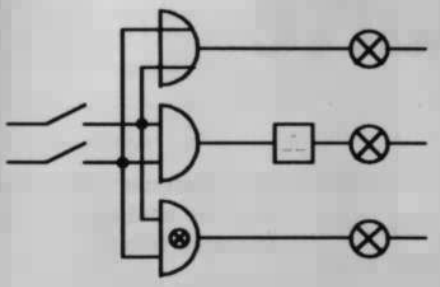
Der Hardware-Aufbau eines Knotens (ADM) zeigt, daß bei geschicktem Einsatz selbst ein so einfacher Prozessor wie der 6802 nicht nur Peripherie „pollen", sondern darüber hinaus ein aufwendiges Leitungsprotokoll aufrecht erhalten kann.

Im zweiten Teil wird auf die noch fehlenden Hardware-Komponenten „Anschluß an den PC" (CCM) und „Erweiterungsmodul" (EXM) eingegangen. Auf der Software-Seite bildet das Omega-Net-Protokoll im Zusammenwirken der einzelnen Komponenten den Schwerpunkt der Beschreibung.

(Wird fortgesetzt)

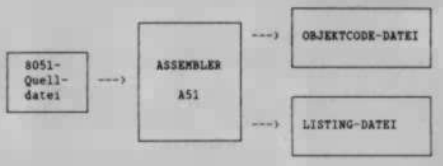
Mit SHAMROCK SOFTWARE können Sie zum Beispiel...

...Steuerungen realisieren,



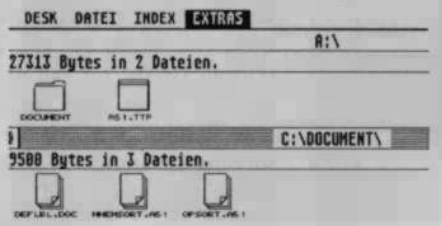
Mit AUTO-65 können ohne maschinensprachliche Kenntnisse Programme für Steuerungs- und Regelungsaufgaben für Einplatinen-Computer mit 65XX-CPU (EMUF) entwickelt werden. Hierzu steht ein Befehlsumfang ähnlich der NC-Sprache STEP-5 von Siemens zur Verfügung. Labels für Sprungziele, Variablen und Konstanten sind möglich, Interrupt-Routinen lassen sich ohne einen Assembler-Befehl programmieren. 6502-Assembler-Mnemonics können aber jederzeit in den Quellentext eingefügt werden. Erzeugt wird der lauffähige Objektcode in Form einer Datei (kann in den Intel-Hexcode konvertiert werden).
Neu: Die Version 1.2 läßt mehr als einen Portbaustein zu! ..... **198 DM**

8051-Programme auf dem PC entwickeln...



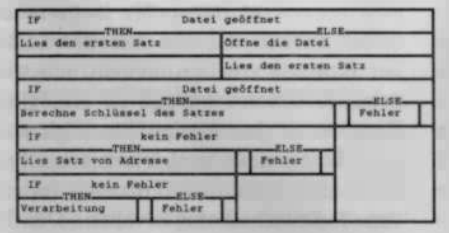
Der A51/PC ermöglicht es, mit einem PC Programme für die Einchip-Microcomputer der Familie 8051 von Intel zu entwickeln. Die Erstellung der Quellendatei ist mit jedem Editor möglich, der reine ASCII-Dateien ohne Steuerzeichen erzeugt. Die Objektdatei kann im Intel-Hex- oder -OMF-51-Format generiert werden. Quellen- und Objektdatei sind kompatibel zum ASM-51 von Intel. INCLUDE-Files können verarbeitet werden, eine Vielzahl weiterer Optionen kann durch Steuer-Anweisungen im Quellentext oder beim Programmaufruf aktiviert werden (Listdatei, bedingte Assemblierung usw.) ..
485 DM

...oder auf dem Atari-ST,



Die Intel-Einchip-Microcomputer der Familie 8051 lassen sich auch mit dem Atari-ST programmieren. Der A51/ST beherrscht alle Befehle und Adressierungsarten sowie die von Intel eingeführten Pseudo-Mnemonics. Er erzeugt aus der Quellendatei, die mit jedem ASCII-Editor erstellt werden kann, eine Listdatei mit eventuellen Fehlermeldungen sowie den direkt ausführbaren Objektcode in Form einer Datei. Dieser kann dann in ein EPROM programmiert oder mit einem Terminalprogramm zum Einchip-Microcomputer übertragen werden ..... **198 DM**

DIN-gerechte Struktogramme erstellen...



STRUKTOR dient zum bequemen Erstellen von Struktogrammen, die für Programm-Dokumentationen unentbehrlich sind. Eingegeben wird nur Text mit dem eingebauten Full-Screen-Editor (Wordstar-ähnlich), die aufwendige Gestaltung der Diagramme übernimmt dann das Programm. Bei erkannten Fehlern wird automatisch der Editor aufgerufen. Die Texte werden abgespeichert, Änderungen sind jederzeit möglich. Das fertige Struktogramm liegt ebenfalls als Textdatei vor und kann in Dokumentationen direkt übernommen werden. STRUKTOR gibt es in zwei Versionen:
STRUKTOR ..... **398 DM**
STRUKTOR junior ..... 98 DM

...und vieles andere mehr:

V.24-OEM, die gepufferte Treiber-Software für COM1 bis COM4, bis 38400 Bd ..... **198 DM**
RS2FILE, speichert RS-232-Daten im Hintergrund auf Diskette ..... **198 DM**
TURBO-DISK-TOOLS, der Turbo-Pascal-Disketten-Monitor ..... **98 DM**
MS-DOS-UTILITY, eine Menge nützlicher Dienstprogramme ..... **98 DM**

shamrock software

Und das ist noch keineswegs alles! Bitte fordern Sie unseren kostenlosen Gesamtkatalog an. (Alle genannten Programme sind auf Industriestandard-kompatiblen MS-DOS-Rechnern bzw. dem Atari-ST lauffähig.)

SHAMROCK SOFTWARE Vertrieb GmbH

Karlstraße 35, 8000 München 2, Telefon 089/595468 + 69

Uwe Hildebrand

Die mc-Transputerkarte

Teil 2: Die Technik

Die mc-Transputerkarte enthält auf einer EURO-Platine einen Transputer vom Typ IMS T414/800, 256 KByte Speicher, der entweder voll mit statischem RAM oder auch zur Hälfte mit EPROM bestückt werden kann, und einen leistungsstarken programmierbaren I/O-Baustein.

Die mc-Transputerkarte kann mit 128 KByte EPROM-Speicher ausgerüstet werden. Deshalb kann die Karte auch im Stand-alone-Betrieb eingesetzt werden. Zur Grundausstattung der Karte gehört ein Monitor in EPROMs, mit dem Download vom Gastrechner auf die Transputerkarte über eine serielle Schnittstelle gefahren werden kann. Mit dem ebenfalls mitgelieferten Cross Assembler kann Software für die Karte auf dem Host-Rechner entwickelt werden. Die Linkstecker auf der Karte sind zu den Steckern nach Inmos signalkompatibel. Die mc-Transputerplatine kann deshalb als Teilsystem in einem größeren Transputersystem eingesetzt werden. In einem solchen System werden üblicherweise Up-, Down- und Subsysteme unterschieden (*Bild 1*).

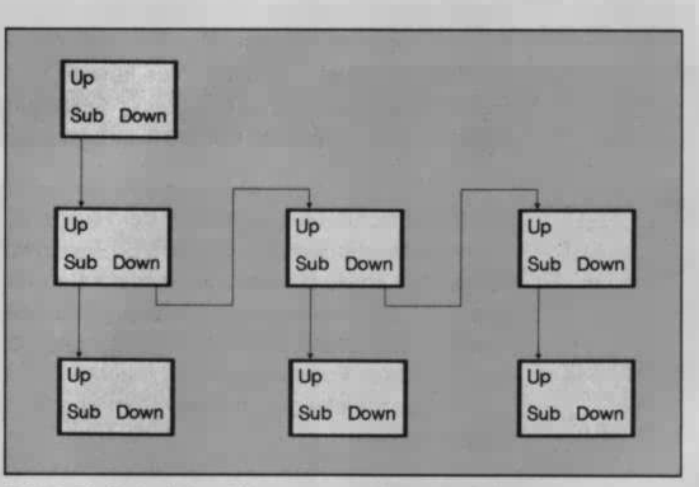


Bild 1. Ein Beispiel zur Vernetzung von Up-, Down- und Subsystemen in einem Multi-Transputersystem

Der Transputer IMS T414

Der Transputer IMS T414 enthält einen 32-Bit-Mikroprozessor, vier Kommunikations-Links, 2 KByte On Chip RAM sowie Speicher- und Peripherieschnittstelle. Der Prozessor verteilt seine Rechenzeit auf die Anzahl der jeweils parallel ablaufenden Prozesse. Von Bedeutung ist hierbei, daß Prozesse, die auf eine Ein- oder Ausgabe warten, und angehaltene Prozesse keine Prozessorzeit verbrauchen. Der Prozessor schließt zwei Timer (einen für jede Prioritätsstufe) mit ein. Bei den vier Kommunikations-Links handelt es sich um bitserielle, bidirektionale Ein- /Ausgabekanäle. Für die Übertragung von Daten aus dem Speicher über die Links verwendet der T414 einen DMA Blocktransfer-Mechanismus. Der Prozessor und die Link-Interfaces können parallel arbeiten, d.h. daß ein Prozeß nicht unterbrochen wird, wenn über die Links ein Datenaustausch vorgenommen wird. Das 2KByte-On-Chip-RAM besitzt eine Zugriffszeit, die der Zykluszeit des Prozessors entspricht. Das 32Bit-Memory-Interface verwendet gemultiplexte Daten- und Adreßleitungen. Es können also 4 GByte linear adressiert werden, wobei das interne RAM am Anfang des Adreßraumes liegt. Ein konfigurierbarer Memory Controller stellt Timing-, Steuer- und Refresh-Signal für eine Vielzahl von Speichersystemen zur Verfügung.

Die Systemsignale

Es gibt einige Instruktionen – hauptsächlich arithmetische und logische Operationen -, die das Error Flag des Prozessors beeinflussen (z.B. bei Overflow). Dieses Fehlerbit kann auch durch Software gesetzt, zurückgesetzt und gelesen werden. Der Zustand des Error Flag wird außerdem direkt auf den Pin

Error

gegeben. Wird der Ablauf eines Prozesses mit niedriger Priorität durch einen Prozeß mit hoher Priorität unterbrochen, so wird der Wert des Error Flag gerettet und das Flag zurückgesetzt. Bei Wiederaufnahme des Prozesses mit niedriger Priorität wird der alte Status wiederhergestellt.

Reset und Boot

Der T414 wird durch einen High-Impuls am Reset-Pin initialisiert, wobei der Pin Analyse auf Low-Pegel zu halten ist. Es folgen die Konfiguration des Memory Interface und die Initialisierung eventuell vorhandener DRAM-Bausteine (Refreshzyklen). Je nach Pegel am Pin „BootFromROM" startet die Abarbeitung an der Adresse #7FFFFFFE (Pin an Vcc) oder der Prozessor erwartet Instruktionen an einem der vier Links (Pin an GND). Im Fall des Power Up Reset muß innerhalb einer festgelegten Zeit die Versorgungsspannung zulässige Werte annehmen und der Oszillator an „ClockIn" schwingen. Danach kann Reset inaktiviert werden.

Pin Analyse

Ein Transputer kann so angehalten werden, daß alle aktuellen Zustände der Prozesse erhalten bleiben, um weitergehende Untersuchungen zu ermöglichen. Dazu ist an den Analyse-Pin High anzulegen. Wenn kein hochpriorisierter Prozeß mehr abläuft, hält der Prozessor innerhalb von 3 Zeitscheibenintervallen. Die Links arbeiten solange weiter, bis eine begonnene Übertragung eines Wortes abgeschlossen ist. Danach ist auch an Reset High-Pegel anzulegen.

Der Adreßraum des IMS T414

Die Speicheradressierung führt der T414 nach den Regeln der Arithmetik für vorzeichenbehaftete ganze Zahlen durch: Die Adresse 0 liegt in der Mitte des Speichers, das 2KByte-On-Chip-RAM beginnt an der negativsten Adresse (#80000000) und endet bei #800007FF. Davon sind die ersten 18 Worte #(80000000 bis einschließlich #80000047) für Systemzwecke reserviert. Der freie RAM-Bereich beginnt danach an der Adresse #80000048, üblicherweise als

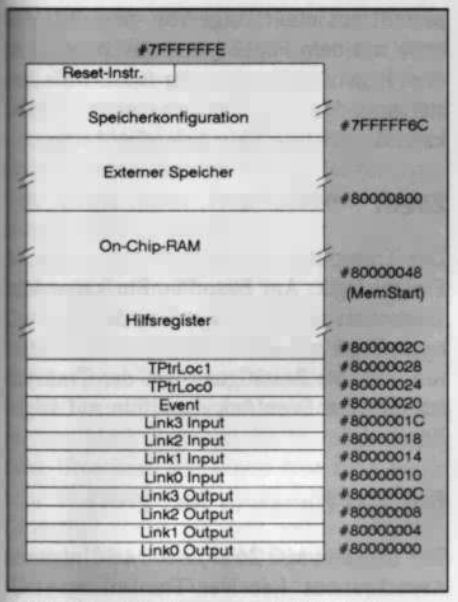


Bild 2. Der Adreßraum des IMS T414

MemStart bezeichnet. Ab Adresse #80000800 liegt der Bereich für das externe RAM. Soll der Transputer aus einem ROM gebootet werden (BootFromROM an Vcc), so wird erwartet, daß dieses ROM ganz oben im Adreßraum liegt. Der Prozessor startet an der Adresse #7FFFFFFE, welche die Reset Instruction enthalten muß. Die Adressen von #7FFFFF6C bis #7FFFFFF8 sind für das programmierbare Memory Interface reserviert. Die Aufteilung des Adreßraumes wird in *Bild 2* dargestellt.

Die Speicherschnittstelle

Wie bereits erwähnt wurde, können über den 32 Bit breiten Adreß-/Datenbus 4 GByte adressiert werden. Neben diesen 32 Leitungen stellt die Speicherschnittstelle noch eine Anzahl von Steuerleitungen bereit. Sie unterstützen den Anschluß von statischem und dynamischen RAM sowie ROM und EPROM. Es sind 13 interne Speicherkonfigurationen einstellbar, wovon eine einfach durch Verbinden zweier Pins selektiert werden kann. Falls keine dieser 13 internen Konfigurationen für eine Anwendung paßt, kann eine externe Konfiguration vorgegeben werden, die praktisch frei gewählt werden kann. Dafür sind die Adressen #7FFFFF6C bis #7FFFFFF8 reserviert. Ein externer Speicherzyklus wird in sechs Abschnitte T1 bis T6, sogenannte Tstates, nach folgenden Regeln unterteilt:
T1: Setup-Zeit für Adressen
T2: Haltezeit der Adressen
T3: Tristate für Lesezyklus oder Setup-Zeit der Daten für Schreibzyklus
T4: Setup-Zeit der Daten ausdehnbar
T5: Daten schreiben oder lesen
T6: Haltezeit der Daten

Die Bus- und Steuersignale der Speicherschnittstelle haben folgende Bedeutungen:

ProcClockOut:
Dieses Signal stellt den internen Prozessortakt zur Verfügung. Im Gegensatz zu dem Signal ClockIn, welches stets eine Frequenz von 5 MHz besitzt, ist der Takt von ProcClockOut vom verwendeten Transputerbaustein abhängig.

MemAD2-31:
Dies sind die Bit 2 bis 31 des gemultiplexten Bus. Da der IMS T414 immer auf ein Wort zugreift, werden die Bit 0 und 1 der Adresse nicht mit ausgegeben. Die Adresse sollte vor Ende von T2 gespeichert werden.

MemnotWrD0:
Während T1 und T2 ist dieses Signal low, wenn es sich um einen Schreibzyklus handelt, sonst besitzt es den Pegel High. Ab T3 wird dieses Signal zum Bit 0 des Datenbus.

MemnotRfD1:
Dieses Signal ist während T1 und T2 low, wenn eine Refresh-Adresse ausgegeben wird. Sonst ist es in diesem Zeitraum high. Ab T3 wird MemnotRfD1 zum Bit 1 des Datenbus.

notMemRd:
Bei einem Lesezyklus wird dieses Signal während T4 und T5 low.

notMemWrB0-3:
Weil der IMS T414 nur Wortadressen ausgibt, sind vier Write-Signale, für jedes Byte eines, vorhanden. Es stehen zwei Schreibmodi zur Verfügung, für welche die Zeit, in der die Schreibsignale aktiv sind, unterschiedlich bemessen ist. Im Fall des „Late Write" sind die Signale während T4 und T5 relevant, im Early-Write-Modus zusätzlich während T3.

notMemS0-4:
Diese fünf Steuersignale sollen den Anschluß verschiedener Speicherbausteine unterstützen. Während notMemS0 fest vorgegeben ist, kann bei den restlichen vier Signalen jeweils eine Flanke zeitlich bestimmt werden, wie in *Bild 3* gezeigt wird.

notMemRf:
Dieses Signal spielt lediglich bei Beschaltung mit dynamischem RAM eine Rolle. Es wird hier nicht näher erläutert.

MemWait: Der Pegel von MemWait wird bei der fallenden Flanke von ProcClockOut abgefragt. Liegt dieses Signal zum Zeitpunkt der Abfrage auf High-Pegel, so wird die Dauer von T4 durch Einfügen von Wartezyklen solange verlängert, bis durch eine Abfrage von MemWait Low-Pegel festgestellt wird. Um den Beginn von T5 stets auf die steigende Flanke von ProcClockOut zu legen, wird nötigenfalls vom Transputer ein zusätzlicher Wartezyklus eingefügt.

DMA über Speicherschnittstelle

Durch die Signale MemReq und MemGranted ist DMA-Zugriff auf den ganzen externen Adreßraum möglich. High-Pegel an Pin MemReq bewirkt, daß die Pins des Adreß-/Datenbusses des Transputers hochohmigen Zustand annehmen und das Signal MemGranted auf High gesetzt wird. MemGranted wird erst dann wieder zurückgesetzt, wenn MemReq deaktiviert wurde. Während eines DMA-Zugriffes können der Prozessor und die Links weiterhin auf das On Chip RAM zugreifen.

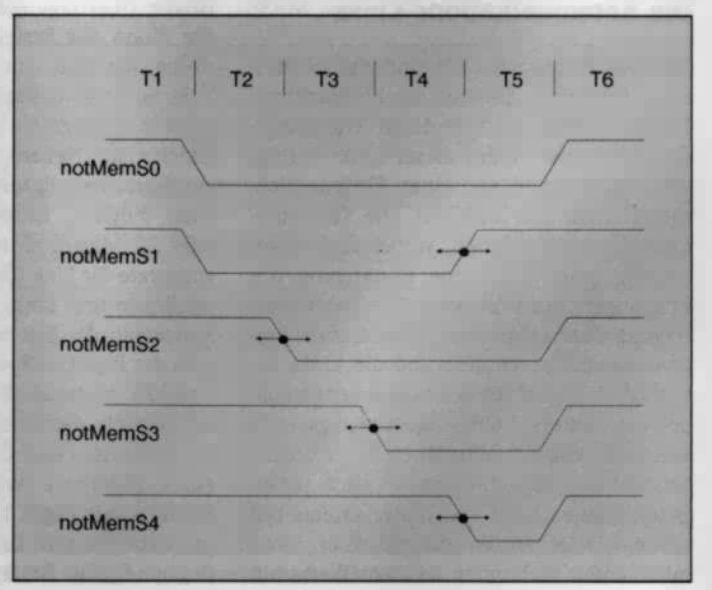


Bild 3. Die Strobe-Signale des IMS T414 sind einstellbar

LinkSpecial	Link0Special	Link123Special	Link0	Link1,2,3
0	0	x	10MBit/s	x
0	1	x	5MBit/s	x
1	0	x	10MBit/s	x
1	1	x	20MBit/s	x
0	x	0	x	10MBit/s
0	x	1	x	5MBit/s
1	x	0	x	10MBit/s
1	x	1	x	20MBit/s

Bild 4. Die Wahrheitstabelle für das Einstellen der Linkgeschwindigkeit

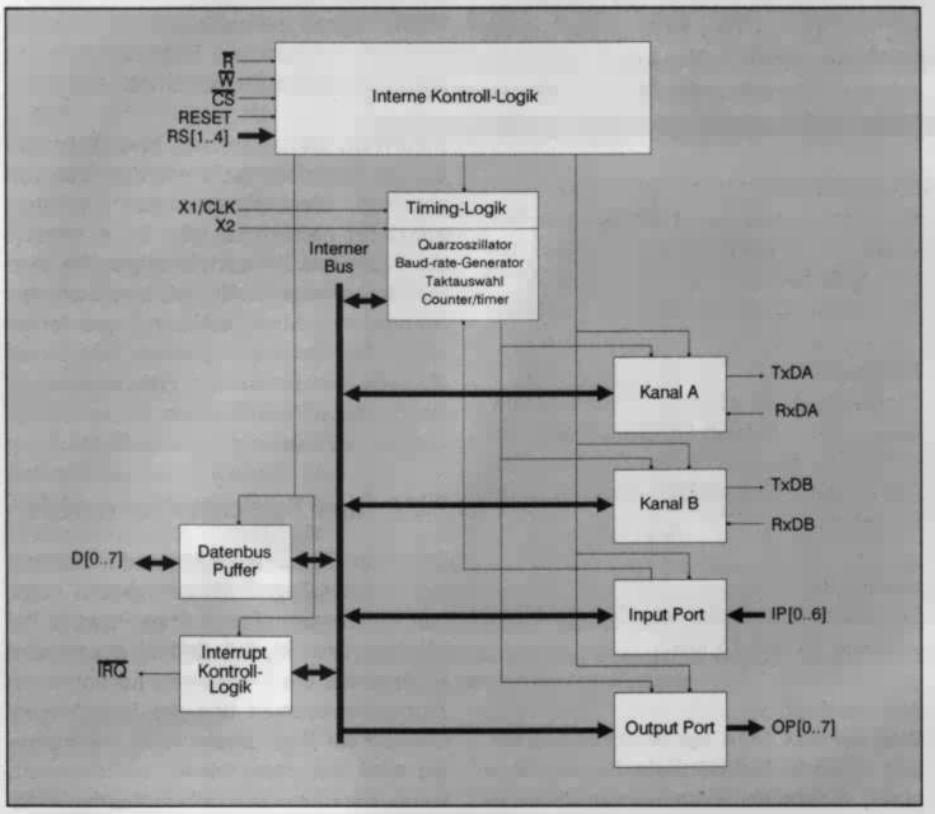


Bild 5. Das Blockschaltbild des I/O-Bausteines MC2681

Die Kommunikations-Links

Auf dem Transputer-Chip sind vier INMOS-Links integriert, die eine direkte Punkt-zu-Punkt-Verbindung zu anderen Transputer-Chips erlauben. Jedes dieser Links verfügt über zwei Anschlüsse, einen für jede Richtung (LinkIn und LinkOut). Die Kommunikation geschieht seriell, wobei nach Übertragung eines Byte eine Bestätigung des Empfängers erwartet wird. Die Datenübertragung über Links kann parallel zum Programmablauf geschehen und die Links haben DMA-Zugriff auf den vollen Adreßraum des Transputers. Da das Übertragungsprotokoll nicht von der Wortlänge des Transputers abhängt (Synchronisation nach jedem Byte), können auch Transputersysteme mit unterschiedlichen Wortlängen über Links miteinander verbunden werden. Weiterhin ist die Übertragung über Links nicht von der Phase des Systemtaktes abhängig: Systeme, die nicht von einem gemeinsamen Takt versorgt werden, können verbunden werden, solange die Frequenz der Systeme gleich ist. Neben der Standardübertragungsgeschwindigkeit von 10 Mbit/s ist es auch möglich, Geschwindigkeiten von 5 oder 20 Mbit/s einzustellen. Die Übertragungsrate für Link 0 kann getrennt von den restlichen drei Links gewählt werden. Das Festsetzen der Datenraten ist von den Pegeln der Pins LinkSpecial, Link0Special und Link123Special abhängig und ist in *Bild 4* tabellarisch dargestellt. Der Ruhezustand der Links ist Pegel 0. Bei der Übertragung eines Datenbyte werden zunächst zwei Startbits mit Pegel 1 gesendet. Darauf folgen dann die acht Datenbit und ein Stopbit (logisch 0). Die Bestätigung des Empfängers besteht aus einer Folge von zwei Bit, das erste mit dem Pegel 1 und das zweite mit dem Pegel 0. Nachdem der Sender die Bestätigung des Empfängers erhalten hat, kann das nächste Byte gesendet werden.

Event

Der Transputer besitzt auch eine einfache Ereignislogik. Am EventReq-Pin kann dem Transputer mitgeteilt werden, daß eine externe Logik mit einem Prozeß kommunizieren will. Als Bestätigung legt der Transputer den Pin EventAck daraufhin auf High-Pegel.

Der I/O-Baustein MC 2681

Der Baustein MC 2681, ein Dual Universal Asynchronous Receiver/Transmitter (DUART), ist in *Bild 5* im Blockdiagramm beschrieben. Die Aufgaben der einzelnen Komponenten des MC 2681, die in Bild 5 gezeigt werden, sollen im Folgenden erläutert werden.

Interne Kontroll-Logik:
Diese Komponente empfängt Kommandos von der CPU und erzeugt daraus die notwendigen Steuersignale für den internen Ablauf. Über die Signale RS1...4 erfolgt die Auswahl der internen Register. Durch das high-aktive Signal RESET kann der Baustein zurückgesetzt werden. Dabei werden interne Register gelöscht, das Output Port (OP 0..7) hochohmig geschaltet, Counter/Timer gestoppt und Kanal A und B inaktiviert. Die restlichen drei Signale ($\overline{CS}$, $\overline{R}$ und $\overline{W}$) dienen zur Festlegung des Zugriffs bzw. der Zugriffsart und sind low-aktiv. Wenn $\overline{CS}$ den Pegel High besitzt, d.h. der Chip nicht selektiert ist, werden die Leitungen D0...7 hochohmig geschaltet.

Timing-Logik:
In dieser Logik sind ein Quarzoszillator, ein Baudraten-Generator, ein programmierbarer 16Bit-Counter/Timer sowie eine Takt-Auswahlschaltung enthalten. Es ist einerseits möglich, einen Quarz von 3,6864 MHz an die Pins X1/CLK und X2 anzuschließen oder aber einen Takt entsprechender Frequenz an X1/CLK anzulegen, wobei X2 unbeschaltet bleibt. Der Baudraten-Generator stellt 18 gebräuchliche Übertragungsraten zur Verfügung, welche im Bereich zwischen 50 und 38,4 KBaud liegen. Der 16Bit-Counter/Timer kann durch geeignete Programmierung verschiedene Eingangsquellen verwenden. Der Ausgang des Counters/Timers steht der Takt-Auswahlschaltung oder an Pin OP3 zur Verfügung. Im Timer-Modus arbeitet der Coun-

ter/Timer als Teiler und kann zur Erzeugung zusätzlicher Baud-Raten benutzt werden. Der Timer gibt einen Takt aus, der die 16fache Frequenz der Baudrate besitzt. Im Counter Modus kann der aktuelle Zählerstand gelesen werden und es ist möglich, den Zähler unter Programmkontrolle zu stoppen und zu starten.

Interrupt Kontroll-Logik:
Der low-aktive Ausgang $\overline{IRQ}$ gibt einen Interrupt nach außen weiter, der intern durch einen von acht Zuständen ausgelöst wurde. Durch Programmieren des IMR (Interrupt Mask Register) ist es möglich die acht Ursachen, die einen Interrupt auslösen, zu maskieren. Durch Lesen des ISR (Interrupt status register) werden alle momentan wirksamen Interruptursachen zurückgesetzt.

Datenbus-Puffer:
Diese Komponente des MC 2681 stellt die Schnittstelle zwischen dem externen Datenbus und dem internen Bus dar. Unter Kontrolle der internen Kontroll-Logik können über die Leitungen D0...7 Schreib- und Lesezugriffe ausgeführt werden.

Kanäle A und B:
Jeder dieser Kommunikationskanäle besitzt einen voll-duplex UART (Universal Asynchronous Receiver/Transmitter). Die Übertragungsfrequenz kann für jeden Receiver und Transmitter getrennt festgesetzt werden. Der Transmitter wandelt die Daten, welche er parallel empfangen hat, in einen seriellen Bitstrom um. Versehen mit Start-, Stop- und (optionalen) Paritybits wird dieser Bitstrom über den Pin TxD des jeweiligen Kanals ausgegeben. In der entgegengesetzten Richtung werden die über den RxD in den Receiver eingelesenen Bitströme in ein Parallelformat umgewandelt.

Input Port:
Die ungelatchten Eingänge IP0..6 können durch einen Lesezyklus aus dem IPR (Input port register) gelesen werden. Dabei wird die Leitung D7 des externen Datenbusses stets auf logisch 1 gesetzt. Es ist auch möglich, die Pins dieses Ports als Hilfseingänge für die serielle Schnittstelle (DUART) zu verwenden.

Output Port:
Das OPR (Output Port Register) hält die Daten, welche über die Pins OP0...7 ausgegeben werden sollen. Wichtig ist dabei zu beachten, daß an den Pins OP0...7 das Komplement des OPR ausgegeben wird. Die Bit des OPR können programmgesteuert einzeln gesetzt und rückgesetzt werden.

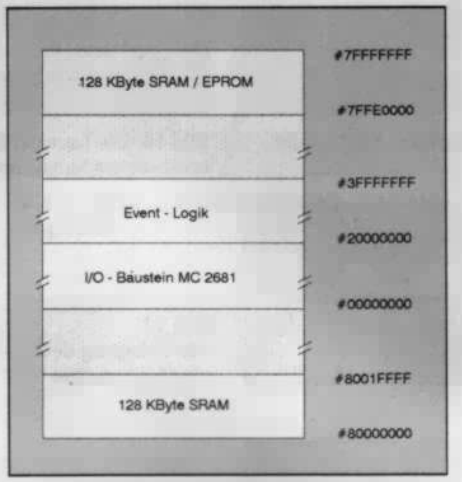


Bild 6. So ist der Adreßraum auf der mc-Transputerkarte aufgeteilt

A31	A30	A29	
0	0	0	MC2681
0	0	1	Event-Logik
0	1	0	-
0	1	1	SRAM/EPROM
1	0	0	SRAM
1	0	1	-
1	1	0	-
1	1	1	-

Bild 7. Die Wahrheitstabelle für die Decodierlogik

Die Ausgangspins dieses Ports können, ähnlich wie dies auch beim Input Port der Fall war, Hilfsfunktionen für die Kommunikationskanäle übernehmen.

Die Aufteilung des T414-Adreßraums

Der Transputer kann auf der Platine die folgenden vier Komponenten ansprechen, die geeignet in den 4 GByte großen Adreßraum des IMS T414 zu legen sind.

1. 128 KByte SRAM
2. 128 KByte SRAM/EPROM
3. I/O-Baustein MC 2681
4. Ereignislogik (Event)

Da für die Decodierung eine möglichst geringe Anzahl von auszuwertenden Adreßbits, d. h. ein möglichst geringer Schaltungsaufwand wünschenswert ist, wird der Adreßraum auf der Platine in acht Segmente zu je 512 MByte unterteilt. Durch diese großzügige Aufteilung des Adreßraumes reichen die drei Adreßbit A31, A30 und A29 zur Decodierung aus. Das *Bild 6* zeigt, wie die vier Komponenten im Adreßraum plaziert sind. Da die Komponenten ausschließlich durch drei Adreßleitungen selektiert werden, tritt innerhalb der einzelnen 512 MByte großen Segmente eine Spiegelung auf. Die Wahrheitstabelle für die Dekodierlogik ist in *Bild 7* dargestellt.

Das Timing

Für den Speicherzugriff des Transputers wurde eine der internen Speicherkonfigurationen des IMS T414 gewählt. Für die Auswahl war entscheidend, daß sowohl auf schnelles SRAM, als auch auf langsamere EPROMs Zugriff möglich sein sollte. Durch Verbinden der Pins MemConfig und MemAD9 des Transputers wird eine Konfiguration festgelegt, die den Anforderungen entspricht. Es ist dabei möglich einen Speicherzyklus in 4 Prozessorzylen durchzuführen oder durch den mitkofigurierten Waitstate Generator einen Speicherzyklus um 3 Zyklen zu verlängern. Bei Einsatz eines IMS T414 mit internem Takt von 20 MHz sieht das Zeitdiagramm für einen Zugriff auf das SRAM wie in *Bild 8* aus.

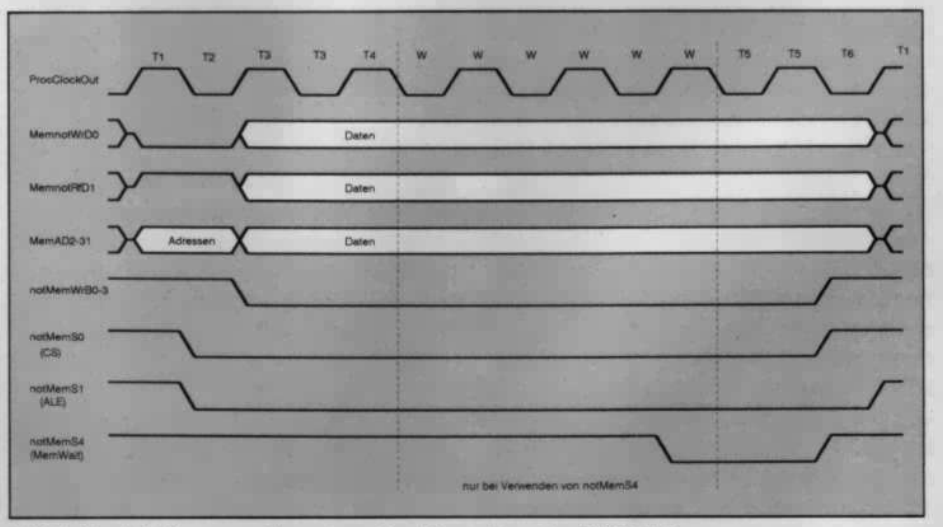


Bild 8. Das Zeitdiagramm für externe Speicherzyklen des IMS T414

INT3	INT2	INT1	INT0	I/O	SS1	SS2	SS3
Bit7							Bit0

Bild 9. Die Bedeutung der Bit für die Interrupteingänge. Ein Null-Bit aktiviert den betreffenden Interrupt

Res3	Res2	Res1	—	Ana3	Ana2	Ana1	—
Bit7							Bit0

Bild 10. Die Lage der Bit für das Rücksetzen und Analysieren angeschlossener Subsysteme; diese kann man einzeln anhalten

Pin2	Pin4	Pin6	Pin8	Pin10
GND	Link Output	Analyse	Reset	Vcc
Link Input	GND	Error	----	Vcc
Pin1	Pin3	Pin5	Pin7	Pin9

Bild 11. Die Belegung der vier Linkstecker

	a	b	c
1	Vcc	Vcc	Vcc
2	O6	O4	O2
3	I2	I6	I5
4	I4	MemWait	INT0
5	INT1	INT2	INT3
6	I3	O3	O5
7	O7	I0	O1
8	I1	O0	-
9	T1OUT	R2IN	R1IN
10	T2OUT	-	-
11	-	-	-
12	-	-	-
13	GND	GND	GND
14	-	-	-
15	MemReq	EventAck	ProcClockOut
16	notMemWrB1	MemGranted	notMemWrB3
17	notMemWrB0	notMemWrB2	notMemRf
18	notMemS3	notMemS0	notMemRd
19	notMemS4	notMemS1	notMemS2
20	MemAD0	MemAD11	MemAD22
21	MemAD1	MemAD12	MemAD23
22	MemAD2	MemAD13	MemAD24
23	MemAD3	MemAD14	MemAD25
24	MemAD4	MemAD15	MemAD26
25	MemAD5	MemAD16	MemAD27
26	MemAD6	MemAD17	MemAD28
27	MemAD7	MemAD18	MemAD29
28	MemAD8	MemAD19	MemAD30
29	MemAD9	MemAD20	MemAD31
30	MemAD10	MemAD21	-
31	-	-	-
32	GND	GND	GND

Bild 12. Die Belegung der VG-Leiste

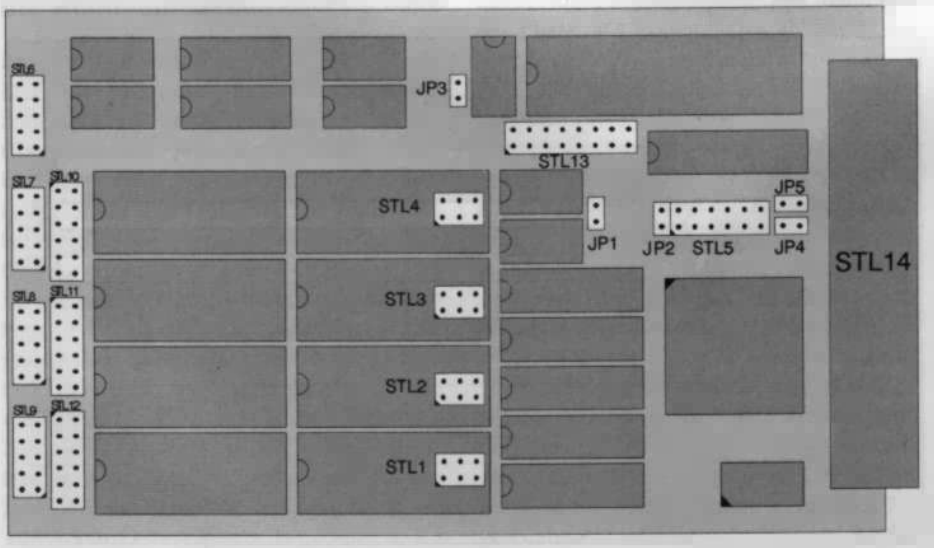


Bild 13. Der Bestückungplan für Jumper und Stiftleisten

Die Event-Logik

Eine der Aufgaben der Event-Logik besteht darin, bei Aktivierung von einem der acht Interrupt-Eingänge einen Interrupt am Pin EventReq des IMS T414 auszulösen. Ein solcher Sammelinterrupt kann durch folgende Zustände ausgelöst werden:

- Aktivierung des Error-Signals eines Subsystems
- Interrupt-Anforderung durch den I/O-Baustein MC 2681
- Low-Pegel an einem der vier freien Interrupt-Eingänge INT0-3. Diese sind auf der VG-Leiste herausgeführt.

Diese acht Interruptsignale können durch einen Lesezugriff auf den Event-Bereich des Adreßraumes über einen Tristate-Treiber auf die untersten 8 Bit des Datenbus geschaltet werden. Dabei bedeutet ein Bit mit Zustand logisch 0 in diesem Byte, daß der entsprechende Interrupt-Eingang aktiviert ist. Die Zuordnung der einzelnen Bits zu den Interrupt-Eingängen zeigt *Bild 9.* Mit einem Schreibzyklus innerhalb des Event-Bereiches des Adreßraumes wird das unterste Datenbyte (D0-7) in ein Latch geschrieben. Von diesen acht Bit werden nur sechs ausgewertet. Drei Bit dienen dazu, für untergeordnete Subsysteme Reset- Signale festzulegen. Die restlichen drei Bits repräsentieren die Analyse-Signale für Untersysteme. Durch Schreiben eines bestimmten Bitmusters in das Latch ist es also möglich, alle Subsysteme einzeln anzuhalten, zu analysieren oder rückzusetzen. Dazu wurde für das unterste Datenbyte die Bedeutung nach *Bild 10* festgelegt.

Die Signale Error, Analyse und Reset von Downsystemen werden auf dieser Karte nicht ausgewertet, sondern lediglich an das Upsystem, d.h. an den Linkstecker für Link0, weitergegeben. Genauso wird das Error-Signal des Transputers dieser Karte über den Link0-Stecker an das Upsystem weitergeleitet. Tritt bei dem IMS T414 auf dieser Baugruppe ein Fehler auf (Pin Error auf High-Pegel), so leuchtet die Fehler-LED; weitere Untersuchungen müssen aber vom Upsystem veranlaßt werden. Dazu werden die Signal Reset und Analyse, die über den Link0-Stecker zur Verfügung stehen, dem Transputer auf dieser Karte an den entsprechenden Pins zugeführt.

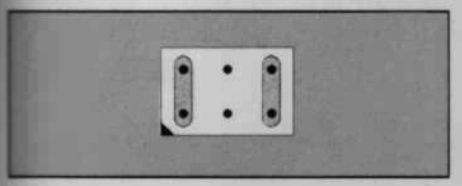


Bild 14. Die Jumperstellung an STL1 ... STL4 für durchgängig RAM

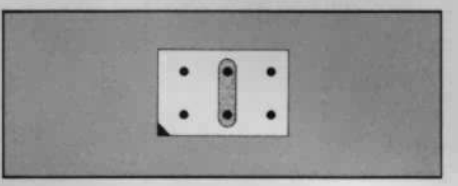


Bild 15. STL1 ... STL4 für EPROMs und RAM eingestellt

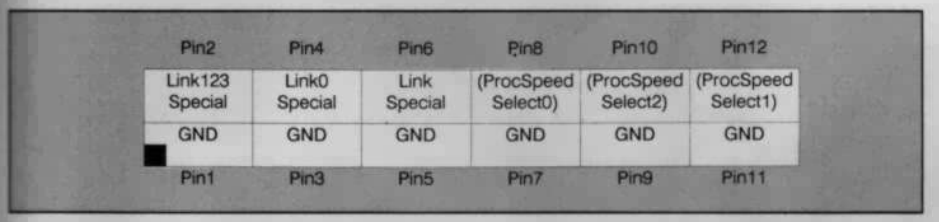


Bild 16. Die Belegung von STL5

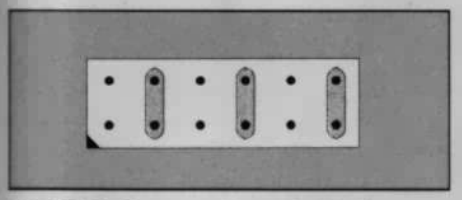


Bild 17. Die Jumperstellung für STL10 ... STL12 für den Anschluß von Subsystemen

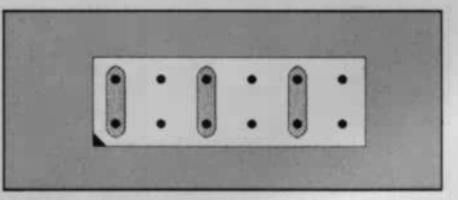


Bild 18. Die Jumperstellung an STL10 ... STL12 für Downsysteme

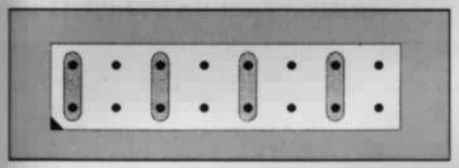


Bild 19. Die Jumperstellung für die Verwendung von I0, I1, O0, O1 als Portleitungen

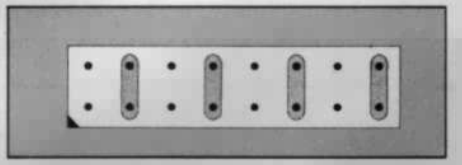


Bild 20. Die Beschaltung bei Verwendung der Signale I0, I1, O0, O1 als Steuerleitung für die RS-232C-Schnittstelle

Steckerbelegungen

Linkstecker:
Die Belegung der Linkstecker ist für alle vier Linkstecker identisch. Diese vier Linkstecker sind im Bestückungsplan der Platine mit STL6, STL7, STL8 und STL9 bezeichnet. *Bild 11* zeigt die Belegung dieser Stecker.

VG-Leiste:
Die Belegung der VG-Leiste (STL14) auf der Transputerbaugruppe ist in *Bild 12* angegeben.

Jumperstellungen:

Bild 13 zeigt die Benennung der Jumper.

JP1:
Der Jumper JP1 muß dann gesetzt werden, wenn die Baugruppe vollständig mit statischen RAM-Bausteinen bestückt ist. JP1 darf nicht gesetzt werden, wenn EPROMs eingesetzt sind.

JP2:
Jumper JP2 ist zu setzen, wenn der Transputer über ein Link gebootet werden soll. Er darf nicht gesetzt werden, falls aus einem EPROM gebootet wird.

JP3:
Dieser Jumper bewirkt, daß die Baugruppe auch ohne ein Upsystem zurückgesetzt werden kann (ohne die Spannungsversorgung abschalten zu müssen). Das Setzen von JP3 hat zur Folge, daß der Pin Reset des Transputers den Zustand logisch 1 annimmt.

JP4:
Wenn der Anschluß MemReq der VG-Leiste (Pin a15) nicht beschaltet wird, so ist durch Setzen von JP4 dieser Pin auf Masse zu legen. Soll eine externe Baugruppe Zugriff auf die Busse des Transputers bekommen, so darf JP4 nicht gesetzt werden. Der Pin a15 der VG-Leiste wird dann direkt auf den Pin MemReq des Transputers geführt, so daß je nach angelegtem Pegel eine Übergabe der Busse des Transputers angefordert wird oder nicht.

JP5:
Der Jumper JP5 muß dann gesetzt werden, wenn der Pin b4 der VGLeiste (MemWait) unbeschaltet bleibt. Benötigt eine externe Logik Wartezyklen innerhalb eines Transputerzugriffs, so darf JP5 nicht gesetzt werden. Auf der Transputerbaugruppe ist die Logik so konzipiert, daß ein High-Pegel am Pin b4 der VG-Leiste sich entsprechend auf den Pin MemWait des Transputers auswirkt.

STL1 ... STL4:
Die Stiftleisten STL1 bis STL4 müssen bei Bestücken der ganzen Platine mit statischem RAM nach *Bild 14* gejumpert werden. *Bild 15* zeigt die Jumperbelegung, wenn EPROMs eingesetzt sind.

STL5:
Auf die Stiftleiste STL5 sind die Anschlüsse des Transputers herausgeführt, die für das Einstellen der Linkgeschwindigkeiten und den Einsatz des IMS T800 notwendig sind. Die Signale an STL5 sind alle über Pull-up-Widerstände auf High-Pegel gezogen (*Bild 16*). Bei Verwenden des IMS T414 müssen stets die folgenden drei Jumper auf der Leiste gesetzt werden: Pin 7-8, Pin 9-10 und Pin 11-12.

STL10 ... STL12:
Über diese Stiftleisten kann eingestellt werden, ob das System am entsprechenden Linkstecker als Sub- oder Downsystem geschaltet werden soll. Die Belegung der drei Stiftleisten ist identisch, wobei STL10 für Link 1, STL11 für Link 2 und STL12 für Link 3 relevant ist. Für ein Subsystem müssen die Jumper nach *Bild 17* gesetzt werden. Die Jumper für ein Downsystem zeigt *Bild 18.*

STL13:
Über die Leiste STL13 kann eingestellt werden, ob die Signale I0, I1, O0 und O1 als Leitungen für das Ein- bzw. Ausgabeport verwendet werden sollen, oder ob diese vier Signale als zusätzliche Steuerleitungen für die seriellen Schnittstellen eingesetzt werden. Die Jumper müssen für den Einsatz als Portsignale nach *Bild 19* gesetzt werden. *Bild 20* zeigt die Belegung für den Einsatz der Signale als RS-Steuerung.

Wird fortgesetzt

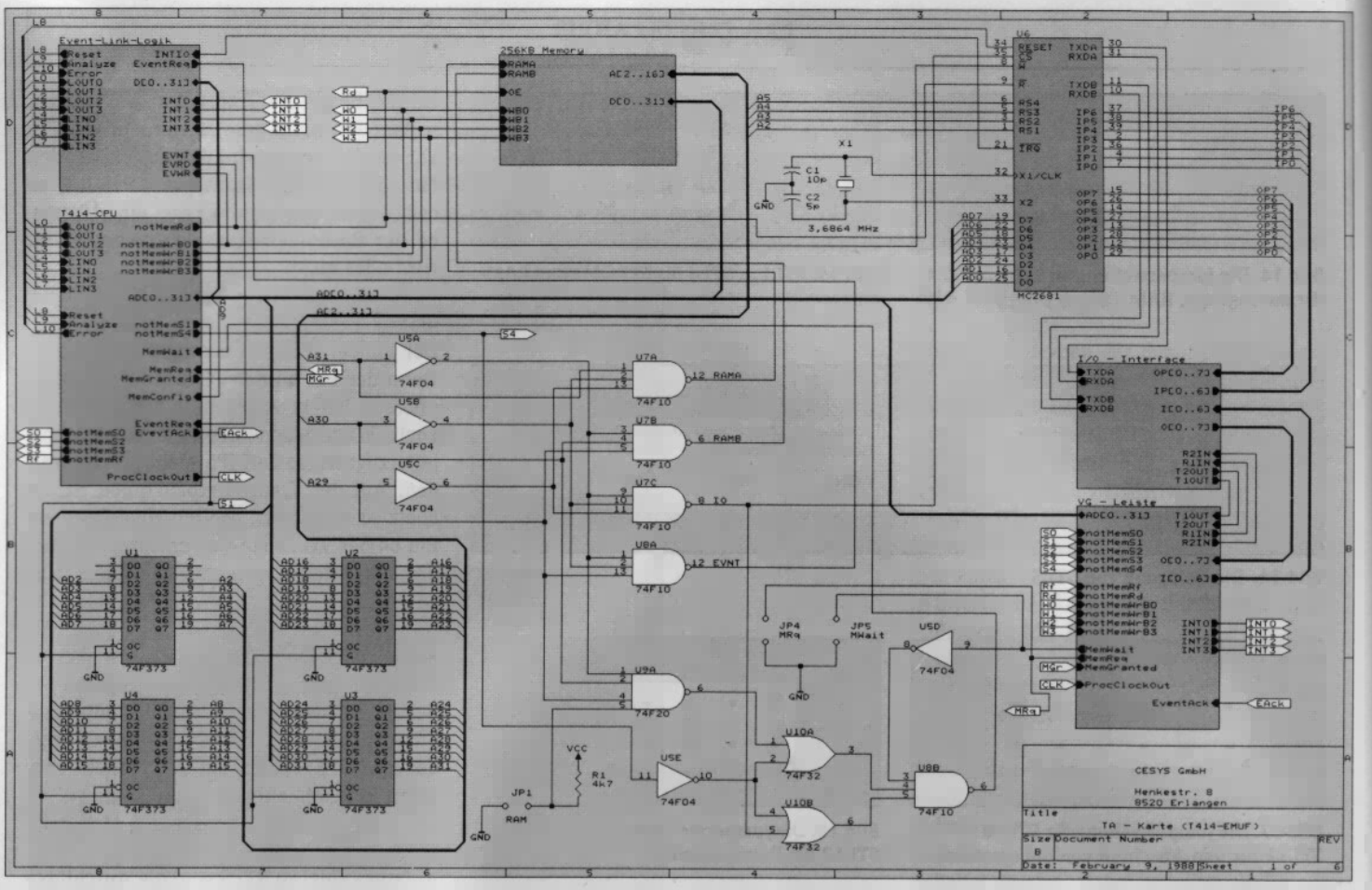


Bild 21. Das Schaltbild in der Übersicht

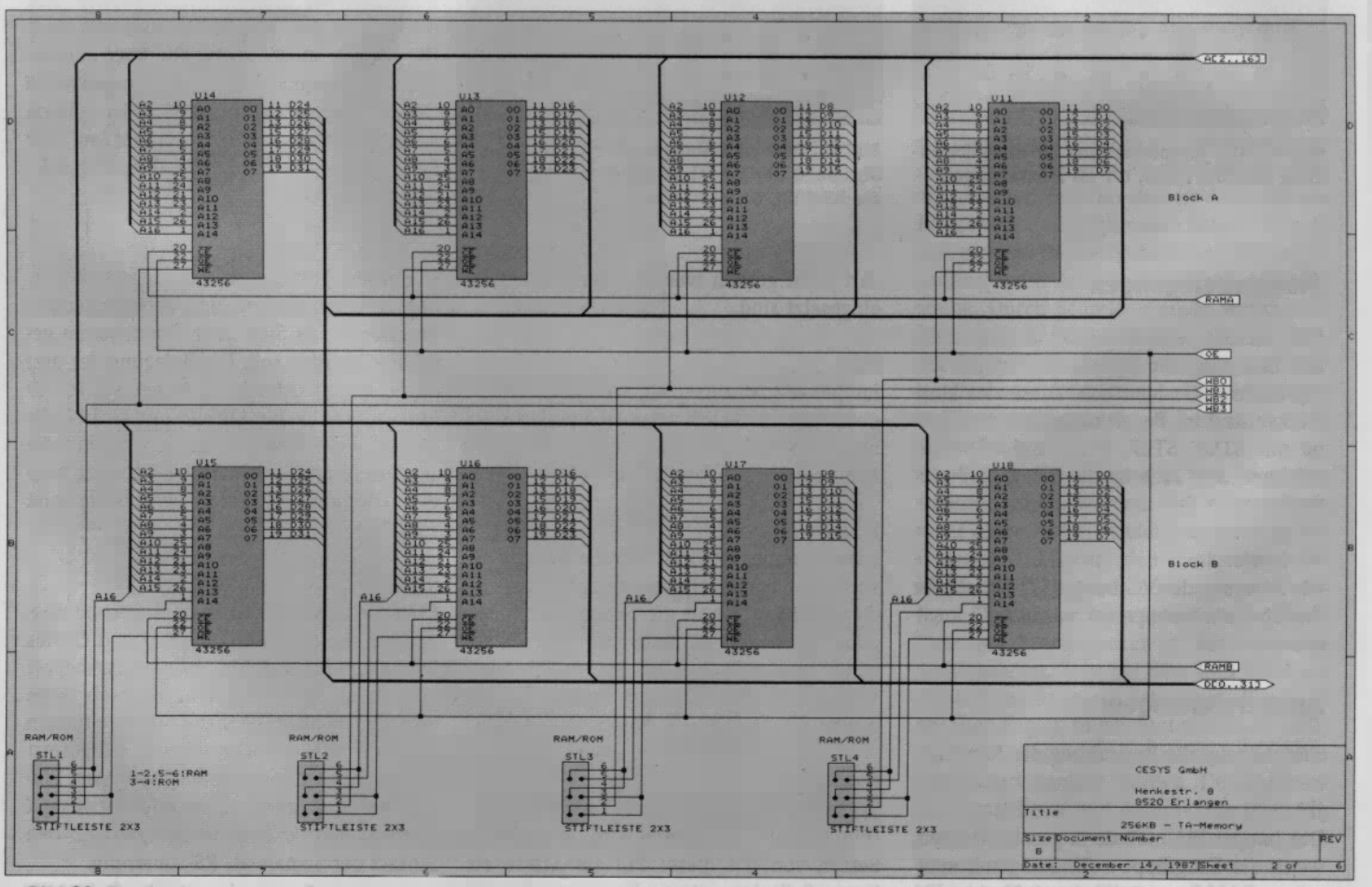


Bild 22. Der Speicher enthält 256 KByte

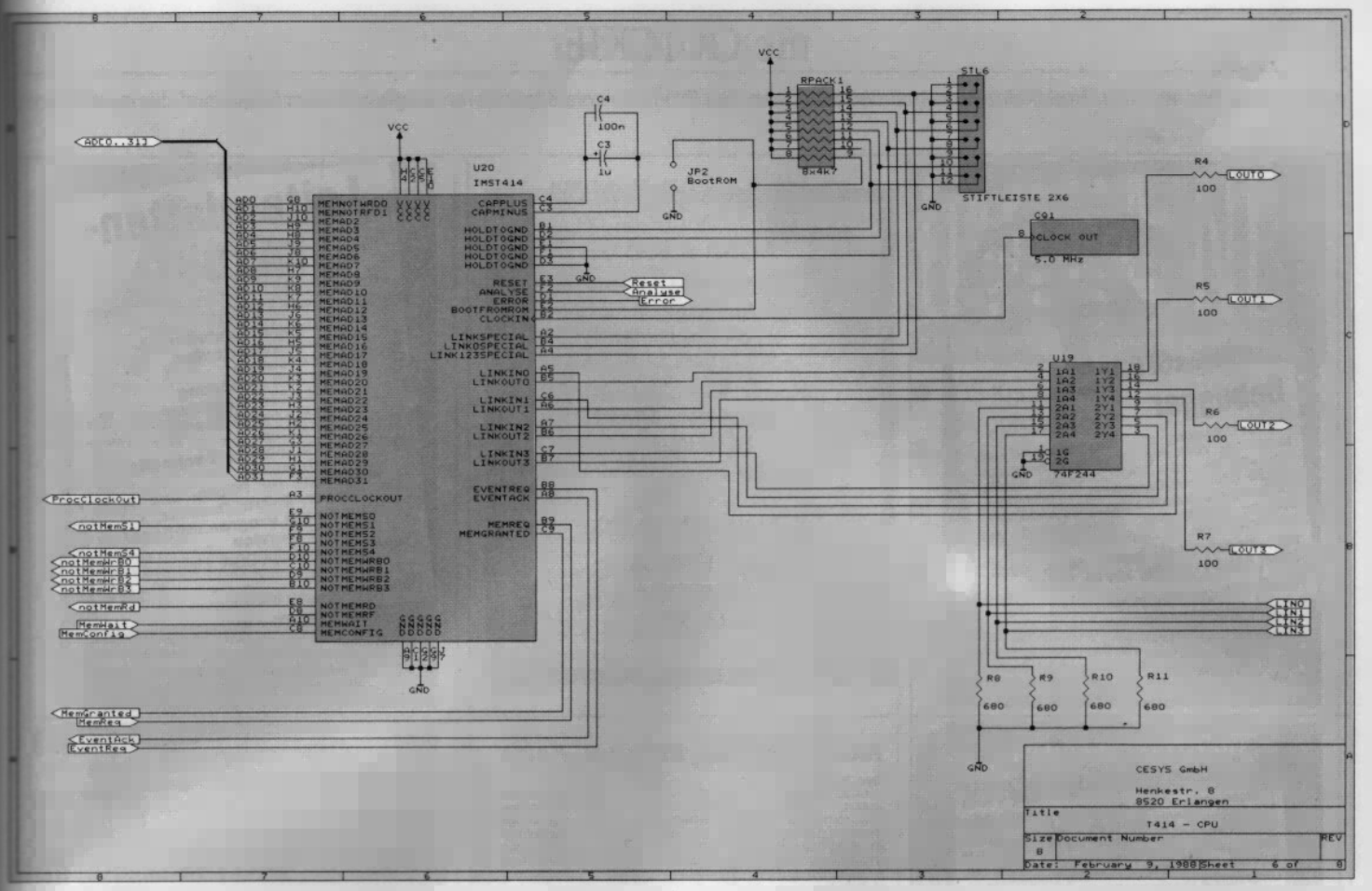


Bild 23. Die CPU mit den Anschlüssen

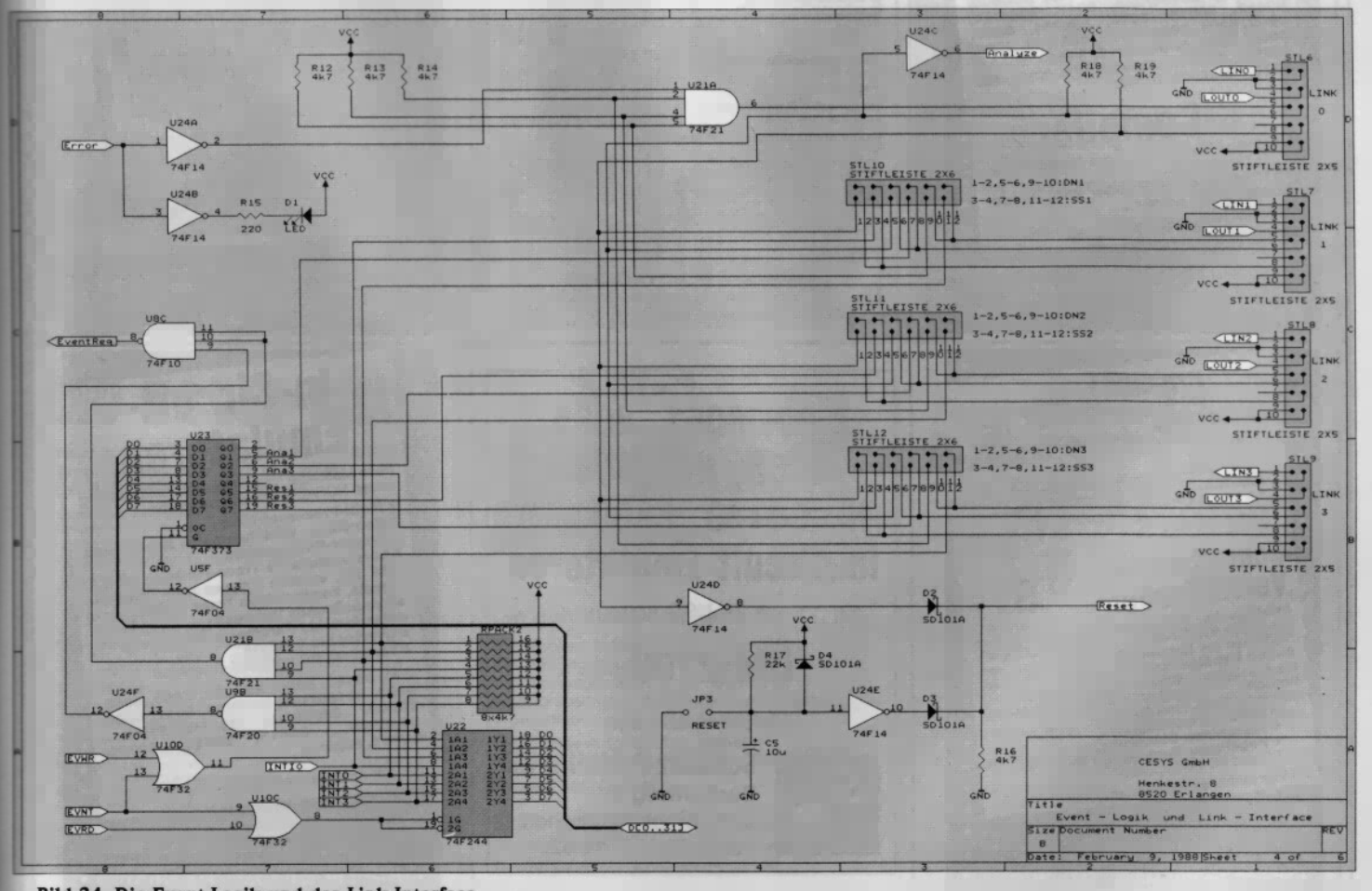


Bild 24. Die Event-Logik und das Link-Interface

mc-QUICKIE

mc-quickies sind aktuelle Produktanzeigen, mit denen Firmen ihre Produkte vorstellen. Verantwortlich für den Inhalt sind die Inserenten.

Simulatoren-Debugger

Screen-Debugger, das preiswerte und wirkungsvolle Werkzeug für Programm-Entwicklung und Test auf XT-/AT-Computern. Phantastische Bildschirm-Darstellung in Fenstertechnik, komplexe Break-Bedingungen, Undo, symbolische Arbeitsweise, Session-Report und zahlreiche Tools, für Zielprozessor-Familien:

Z80, 8085, 8048, 8051, 6801, 6805, 6809, 68HC11, 6502, 32010, 32020

Händler-Anfragen willkommen.

Ingenieurbüro für Datenverarbeitung
Dipl.-Ing. Hans-Jürgen Wilke
Adalbertsteinweg 26 D-5100 Aachen
Tel.: 0241/542228 FAX: 0241/533230

Der Scanner ist da !
AGFA S 800 GS
Flachbett - Scanner
800dpi Auflösung in 12 Stufen
64 Graustufen
DIN A4
Anschließbar an
Macintosh, IBM PC
SCSI Interface
Optional RS232, Centronics

Info anfordern !

Preis : DM 10.990.-

Siegfried Theimer GmbH
Rohlerstr.10, 6484 Birstein 6
Tel. 06054/8030 Fax. 06054/2619 Tlx. 4184314

Leiterplatten-LAYOUT

Professionelles Leiterplatten-Entwicklungssystem für PCs:

- **Schaltplan-Erstellung**
- **Interaktives Routen**
- **Auto-Routing**
- **1/1000 Zoll Auflösung**
- **1–8-Lagen-Multilayer**
- **480x810-mm-Boards**
- **SMD-fähig**
- **Komfortable und schnelle Arbeitsweise**
- **Outputs für Photo- und Stiftplotter, Matrix-Drucker**
- **Laufende Programmpflege/ Update-Service**

Demo-Version (CGA/EGA, Mouse) 30.– DM

Händler-Anfragen willkommen.

Ingenieurbüro für Datenverarbeitung
Dipl.-Ing. Hans-Jürgen Wilke
Adalbertsteinweg 26 D-5100 Aachen
Tel.: 0241/542228 FAX: 0241/533230

Cross-Assembler

Professionelle Cross-Assembler für PC-DOS/MS-DOS und VAX-Systeme, MACROS, Linker, Library-Manager, Conditionals, unbegrenzte Programmlänge, verschachtelte Strukturen, schnelle Compilation, zahlreiche Zielformate.

Unterstützte Prozessor-Familien:

Z8, Super-8, Z80, Z180, 64180, Z280, Z8000, Z80.000, 8080, 8085, NSC800, 8088/86, 80186/286, 8048, 8044/51, 80452, 80515, 8096, 83c351, 8400, 1802, 6301, 6800/02/08, 6801/03, 6804, 6805, 6809, 68000/8/10, 68020, 65c01/02/11, 65c816, 32000, V20/V30, F8/3870, NEC7500, NCR32

C-Compiler

Vollständige Version mit Cross-Assembler/Linker für Zielsysteme:

8051, 6801, 6301, 68020, Z80, 64180, NSC800, 6809, 68HC11

Händler-Anfragen willkommen.

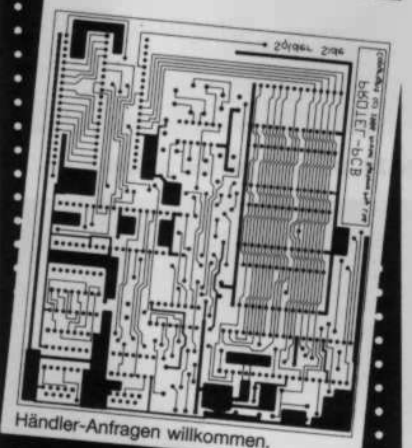
Ingenieurbüro für Datenverarbeitung
Dipl.-Ing. Hans-Jürgen Wilke
Adalbertsteinweg 26 D-5100 Aachen
Tel.: 0241/542228 FAX: 0241/533230

RAMs EPROMs EPROMs RAMs

Dyn. RAMs		
4164-20 (200 ns)	HITACHI	Auf Anfrage
4164-12 (120 ns)	FUJITSU	Auf Anfrage
4164-10 (100 ns)	SAMSUNG	Auf Anfrage
41256-15 (150 ns)	TOSHIBA	Auf Anfrage
41256-12 (120 ns)	TOSHIBA	Auf Anfrage
41256-10 (100 ns)	FUJITSU	Auf Anfrage
41256-07 (70 ns)	NMB	Auf Anfrage
Stat. RAMs		
µPD 4016 C3 (150 ns)	NEC	DM 4,23
HY 6264 LP10 (100 ns)	HYUNDAY	DM 9,80
µPD 43256-12L (120 ns)	NEC	DM 24,51
EPROMs		
2764-25	TEXAS INSTR.	DM 9,46
27256-20	NEC	DM 14,25
CMOS-Mikroprozessoren		
V20 (8 MHz)		DM 15,79
V30 (8 MHz)		DM 21,57
Numerische Coprozessoren		
8087 - 80287 - 80387		Auf Anfrage
Festplatten-Laufwerke von Seagate		
ST 225 (21,4 MB, 65 ms)		DM 503,90
ST 238R (32,7 MB, 65 ms)		DM 522,12
ST 238R inkl. Controller		DM 695,40
ST 251 (42,8 MB, 40 ms)		DM 818,52
ST 125 (21,4 MB, 35 ms)		DM 652,08
ST 157R (49,1 MB, 30 ms) inkl. Controller		DM 1064,76
ST 4096 (80 MB, 28 ms)		DM 1502,52

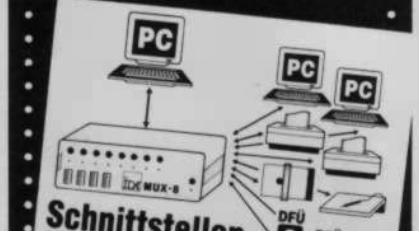
Angebot freibleibend!

MEMORY ELECTRONICS
Inh. Ursula Nohe, Dechsendorfer Straße 10,
8522 Herzogenaurach, Telefon 0 91 32/6 11 61



Händler-Anfragen willkommen.

Ingenieurbüro für Datenverarbeitung
Dipl.-Ing. Hans-Jürgen Wilke
Adalbertsteinweg 26 D-5100 Aachen
Tel.: 0241/542228 FAX: 0241/533230



Schnittstellen-Vervielfacher

Intelligente Schnittstellen-Vervielfacher der Serien **VARIO-MUX** und **MULTI-MUX** ermöglichen die **komfortable Bedienung von 4–256 Kanälen** durch PC oder Mini. Flexibelste Ausstattung, Schnittstellen-Konversion, galvanische Trennung, Daten-Vorverarbeitung, zahlreiche Modi, ideal für:

- **Daten-Erfassung,**
- **Maschinen-Steuerung,**
- **Regelungs-Technik**

mit PC und Mini-Computern.

Händler-Anfragen willkommen.

Ingenieurbüro für Datenverarbeitung
Dipl.-Ing. Hans-Jürgen Wilke
Adalbertsteinweg 26 D-5100 Aachen
Tel.: 0241/542228 FAX: 0241/533230

Z-80, 8051, 8085, 8088, NSC 800, HD-64180, 68000/10 In-Circuit-Emulatoren

z. B. NSC 800

DM 2100,–

Lieferung ab Lager

S+M

Schwarz & Müller KG
Buchenweg 5
8209 Stephanskirchen
Tel. 0 80 31/7 11 62

In-Circuit-Emulatoren

Neue Maßstäbe hinsichtlich Preis und Leistung setzen NORAL-In-Circuit-Emulatoren:

- **Multi-Prozessor-fähig**
- **Komplexe, verschachtelte Break- und Trigger-Bedingungen**
- **Leistungsstarke Trace-Funktion 32 K x 56 Bit, externe Trigger und Trace-Signale, komfortables Trace on/off**
- **Symbolisches Debuggen**
- **Komfortable Parameter-Einstellung**
- **Overlay bis 1 MByte**
- **Terminal, PC und VAX-Anschluß**

Z80, 64180, 8085, NSC800, Z280, 8088, 8086, 80188/186, V20/V30, 6502, 6809, 68000, 68008/10, 8031/32/51/52..., 8048/49/..., 80515, 80512, 80535, 80451, Super-8

Händler-Anfragen willkommen.

Ingenieurbüro für Datenverarbeitung
Dipl.-Ing. Hans-Jürgen Wilke
Adalbertsteinweg 26 D-5100 Aachen
Tel.: 0241/542228 FAX: 0241/533230

mc-QUICKIE

mc-quickies sind aktuelle Produktanzeigen, mit denen Firmen ihre Produkte vorstellen. Verantwortlich für den Inhalt sind die Inserenten.

mtr3, DER EPROM-HELFER
sekundenschnell — automatisch — handlich

- für EPROMs und EEPROMs bis 256Kbit (Option 1Mbit),
- Adapter selbst für wenig gebräuchliche Bauelemente,
- sekundenschnelle aber auch konventionelle Programmierung,
- Typwahl automatisch oder zumindest ohne Datenblatt-Hilfe,
- Emulatorfunktion
- V.24/RS232C-Schnittstelle, einfach konfigurierbar,
- Netz- und Akkubetrieb,

und das alles im kleinsten (189 x 138 x 48) autonomen EPROM-Programmierer für

DM 2257,20 Sfr 1950,— exkl. Wust

Meßtechnik Dr.-Ing. Ranfft
Dörpfeldstr. 15, D-5657 Haan 2, Tel. 0 21 04 / 6 28 27

Hedinger + Partner KG
Zentralstr. 3, CH-8610 Uster, Tel. 01 / 940 90 55

C-COMPILER

MI-C für CP/M, CP/M 86, MS-DOS

MI-C vereint hohen Bedienungskomfort mit hervorragender Leistung

- Vollständige Version mit 13stelliger BCD-Arithmetik für Gleitkommazahlen
- Erzeugt kurze und schnelle ROM-fähige Programme
- Ausgabe im Assemblercode des Zielprozessors
- Kompatibel zu MAC80/L80 (MASM/LINK) v. Microsoft
- Für 8086: 4 Speichermodelle/8087-Co-Prozessor
- Fehlerverfolgung mittels Trace möglich
- Umfangreiche Bibliothek (AMD9511-Paket erhältlich)
- UNIX-kompatibel
- Deutsche oder englische Version lieferbar
  8"-/5,25"-/3,5"-/3"-Disk + dt. Handbuch

MI-C für CP/M	445.– DM
MI-C für CP/M 86 oder MS-DOS	575.– DM
MI-C-Crosscompiler (Ziel 8080/Z80)	745.– DM
MI-C-Crossassembler + Linker (Ziel 8080/Z80)	645.– DM
MI-C-Crosscompiler/-assembler (Ziel 8051)	1495.– DM
MI-C-Crosscompiler/-assembler (Ziel 8096)	1495.– DM
MI-C-Crossassembler (Ziel 68000/10/20)	795.– DM
MI-C-Crossassembler + Linker (Ziel Z280)	795.– DM

Herbert Rose, Bogenstraße 32, 4390 Gladbeck, Telefon 0 20 43/2 49 12 und 4 35 97
Vertrieb in Österreich:
Dr. Willibald Kraml, Microcomputer-Software, Degengasse 27/16, A-1160 Wien

---

OMEGA-NET

das vielseitige Peripherie-Netz für PCs

- 255 Knoten mit Mikroprozessor
- 500 m Knotenabstand
- 20 KB Assembler-Source-Bibliothek
- Anschluß an serielle Schnittstelle
- **Preis ab 700 DM**

OMEGA-LINK
Neumarkt 1a
5000 Köln 1
Telefon
02 21-24 87 52

Preis-Sensation

DIN-A3-Plotter

mit Papierbewegung

TSS 860

HP-GL-kompatibel

DM 2900.–

6 Farben
0,025 mm Auflösung
400 mm/s Zeichengeschwindigkeit
Centronics- und V.24-Interface
56 Zeichenbefehle

Lieferung per Nachnahme

TSS-Schmitz
In der Holl
5223 Bierenbachtal · Tel. 0 22 93/21 88
Inh. Brigitta Schmitz

mc 5/88

HARDWARE-MESSWERTERFASSUNG

– IBM XT/AT – CBM – hier einige Auszüge IBM –

	ab
IEEE-488-(IEC-Bus-)Platine und Software	DM 490
32-Bit-Optokoppler-Input-Platine	DM 490
32-Bit-Optokoppler-Output-Platine	DM 490
8-Bit-32-Kanal-A/D-Wandler, 8 µs Wandlung	DM 590
8-Bit-32-Kanal-A/D-Wandler mit RS-232-Schnittstelle	DM 620
12-Bit-32-Kanal-A/D-Wandler, 2 µs Settling time	DM 890
12-Bit-4-Kanal-D/A-Wandler, 1 µs Settling time	DM 590
72/192-Bit-Input-/Output-Platine, 8/4 bitweise programmierbar	DM 350/DM 540
Relais + I/O-Platine (12 Relais 1× um, 220 VAC, 3 A/60 W + 12× TTL-I/O)	DM 560
8fach-RS-232-Umschaltplatine (Umschaltung programmierbar od. Alt+1..8)	DM 790
Multiboard 86, 12 Bit–32 K A/D + 12 Bit–2 K D/A + 96 TTL-I/O, gesockelt	DM 1675
Thermoboard 86, –50 °C–+150 °C, 16 Kanäle, od. d. –50 °C–+1150 °C	DM 980
Programmierbarer Timer-Counter, 9fach/24fach, 16 Bit	DM 350/DM 680

NEU IM PROGRAMM

Elektrostatische Plotterauflagefolie zum Festhalten des Plotterpapiers	DM 295
12-Bit-A/D-Turboboard, 16 µs, mit 128 KByte RAM	DM 1850
12-Bit-A/D-Turboboard, 16 µs, mit 128 KByte RAM	DM 2350
Logic-Analyzer-Card, 30 MHz/50 MHz/100 MHz	DM 980 – DM 1453 – DM 2147
32-Bit-Open-Kollector-Output-Platine, max. 50 V/500 mA	DM 390
PT-100-Verstärker für versch. Bereiche, 100 °C, 200 °C, 800 °C	DM 150
16-Bit-8-Kanal-A/D-Wandler (Dual-Slope)	DM 920
Einstellbare Quarzzeitbasis	DM 280
Opto IN/OUT je 32 Bit	DM 790
Freiprogrammierbarer Funktionsgenerator	DM 750
RS232 16-Bit-Out-32-Bit-Input TTL	DM 590
RS232 12-Bit-32-Kanal ADC 25 µs	DM 940

Info kostenlos!

LOTHAR BOCKSTALLER
Hard- und Software – Hadwigstr. 16, 7867 Wehr 2, Telefon 0 77 61/18 08

PRINT TECHNIK

VIDEOTEXT-DECODER
C 64/128 + AMIGA DM 298,–,
für ATARI, IBM DM 398,–

VIDEOTEXT ist ein Informationsservice, der gleichzeitig mit TV-Programmen von ZDF, ORF, SRG, SKY, RAI etc. ausgestrahlt wird. Mit einem dafür geeigneten Fernseher kann man aktuelle Nachrichten, Sport, Börsenberichte, Flugverbindungen, Wetterberichte etc. abrufen. ABER AUCH DAS VIDEORECORDER-TUNER-SIGNAL kann zum Empfang der Informationen mittels des C 64/128er-Computers genutzt werden. Hard- und Software erlauben VIDEOTEXT seitenweise aufzurufen, auszudrucken, auf Diskette zu speichern. Eine sensationelle Neuentwicklung der Print Technik, Wien, für den C 64/128-User.

C 64/128-VIDEODIGITIZER DM 248,–
Auch für ATARI und IBM erhältlich.

ATARI-Scanner inkl. DTP DM 1998,–
IBM-Scanner inkl. DTP DM 2598,–

ÖSTERREICH · 1060 WIEN · STUMPERGASSE 34 · TEL. 02 22/5 97 34 23 · TELEX 1 12 996

SCHWEIZ · TEL. 0 32/87 24 29
HOLLAND · TEL. 0 79/41 25 53

8000 München 40 · Nikolaistr. 2 · Tel. 0 89/36 81 97
Katalog DM 3.– · Tägl. Versand · Telex 5 23 203 d

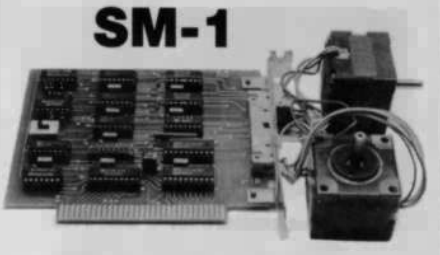
MC BASIC-EMUF

Neu: mc-GAL-Programmer aus mc 2/88, programmiert GAL 16V8 und 20V8 mit BASIC-EMUF oder IBM-PC

Bausatz GAL-Programmer o. TEXTOOL	DM 98,–
Fertigplatine GAL-Programmer o. TEXTOOL	DM 148,–
3M-TEXTOOL-Sockel-Modul, 24polig	DM 49,–
Fertiggerät im Gehäuse für IBM-PC	DM 348,–
Basic-EMUF	
BS1: Platine, GAL, deutsches Manual	DM 98,–
BS2: BS1 + IP80C52, XTAL1	DM 196,–
BS3: RTC58321, AKKU	DM 29,–
BS4: µPD7002, ZN458, XTAL2	DM 29,–
BS5: ULN2003A = 75468, 2× Relais	DM 29,–
BS7: BS1 + TL7705, MAX232, IP80C52, 74HCT08, 74HCT373, 71055, RAM8K	DM 238,–
BSLCDKEY: Bausatz komplett	DM 198,–
FPLCDKEY: Fertigplatine komplett	DM 248,–
Basic-EMUF-Fertigplatine, komplett mit 32 kB RAM	DM 638,–
Basic-EMUF-Fertigplatine, Lochrasterversion mit 8 kB RAM	DM 438,–
PASCAL-CROSS-COMPILER (MS-DOS)	DM 969,–
CROSS-ASSEMBLER (MS-DOS)	DM 869,–
CROSS-SIMULATOR (MS-DOS)	DM 948,–

Intec Electronic GmbH
Rheingrafenstraße 37, 6501 Wörrstadt
Telefon 0 67 32/50 29

SM-1



Die Schrittmotorenkarte für XT, AT

- 2 bipolare Schrittmotoren anschließbar
- 8-Bit-Eingabeport (TTL-Pegel)
- 6-Bit-Ausgabeport (o. K. 500 mA/50 V)
- Motorspannung intern/extern umschaltbar
- Stromreduzierung für Ruhezustand

Steckkarte aufgebaut u. getestet mit 2 Motoren 3,6°/1,8° 12 V DM 189,–

Versand per Nachnahme

Soft- und Hardware Tetzlaff
Grotenbachstraße 9, 4600 Dortmund 50
Telefon 02 31/71 30 42 ab 15.30 Uhr

TRIO

Fakturierung, Mahnwesen und Buchhaltung

Das komplette Paket für Selbständige und Kleinunternehmen:

- 3 Funktionen in einem Programm, auf einer Disk
- einfache und sichere Benutzerführung, ausführliche Bedienungsanleitung (130 Seiten)
- **Interface für DATEV, dBase usw.**
- Artikelverwaltung, Kundenverwaltung, Serienbriefe, Umsatzstatistik, Rechnungsbuch, Creditoren, Debitoren, Kontenblätter, Betriebsübersicht, Journal, G&V, Bilanz
- max. 32 000 Kunden, Artikel und Buchungen
- **14 Tage frei zur Ansicht**

Der Buchhaltungsteil trägt das Prüfsiegel „anwenderfreundliche Software" und wurde von CHIP, C't, Softwaretestjahrbuch, PC+SOFT, hervorragend beurteilt.

780.– bis 980.– DM für XT/AT/PC

Informationen bei:

RÖNTGEN SOFTWARE
Oltmannsstr. 34, 7800 Freiburg, Tel. 07 61/40 87 40

mc-QUICKIE

mc-quickies sind aktuelle Produktanzeigen, mit denen Firmen ihre Produkte vorstellen. Verantwortlich für den Inhalt sind die Inserenten.

PLATINEN-LAYOUT

Leiterplatten-Entwurf auf PC/XT/AT-Systemen

BOARDSTAR ist ein Software-Paket für Entwurf und Reinzeichnung gedruckter Schaltungen und bietet Ihnen professionelle Leistungsmerkmale zu einem fairen Preis. Komfortable mausgesteuerte Benutzeroberfläche - Leiterbahnführung im 1/40° Raster - Bearbeitung doppelseitiger Platinen bis 516 qcm - Verwaltung von Macro-Bibliotheken - Reinzeichnung mit jedem handelsüblichen Plotter.

ROUTSTAR ist der Autorouter zum BOARDSTAR und erstellt selbsttätig Layouts aus vorgegebenen Schaltungsbeschreibungen: Layout-Beschreibung durch Grafik- oder Textfiles - Berücksichtigung vorgegebener Designkriterien - Problemlose Verarbeitung von BUS-Strukturen.

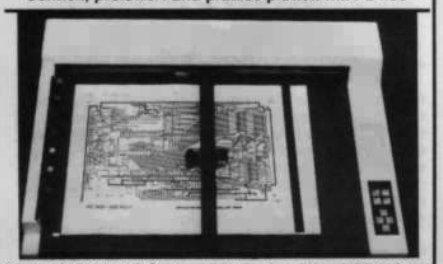
BOARDSTAR (interaktiver Layout-Editor)	698,- DM
ROUTSTAR (Autorouter zum Layout-Editor)	698,- DM
BOARDSTAR-Paket (enthält BOARDSTAR + ROUTSTAR)	998,- DM
Lauffähiges Demo-Paket (wird angerechnet)	70,- DM
DIN-A3 Flachbett-Plotter PL-450 (8-Stift/HPGL/0,025mm)	2450,- DM

Wir bieten Ihnen auch ein breites Spektrum von Service-Leistungen rund um das BOARDSTAR-Programm, vom Plot- u. Repro-Service bis zur Platinenfertigung nach angelieferten BOARDSTAR-Disketten. Fordern Sie Informationsmaterial oder besser gleich das Demopaket an.

Datentechnik Dr.Gert Müller Diezstraße 2A
D-5300 Bonn 1
Tel. 0228/217297

TURBO-PLOTTER

Schnell, preiswert und präzise plotten mit PL-450



Format bis DIN-A3, HP-GL* kompatibel, serienmäßig mit RS232C und Centronics-Schnittstelle ausgerüstet, Pen-Speed maximal 565 mm/s, programmierbarer Stiftwechsel für 8 Farben (Pens), Aufnahme für HP-Standard Faser-/Keramik-/Tusche-Pens, Pen-Magazin mit Automatic-Verschluß, Auflösung 0,025 mm, verschied. Zeichensätze und und.....

Sofort lieferbar... Bitte erfragen Sie den aktuellen Preis!

Datentechnik Dr.Gert Müller Diezstraße 2A
D-5300 Bonn 1
Tel. 0228/217297

* HP-GL ist ein Warenzeichen der Hewlett-Packard GmbH

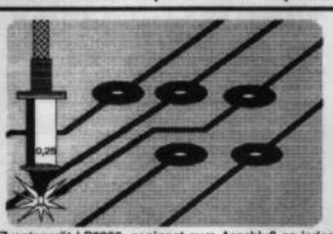
Info anfordern !

Der Drucker ist da !
AGFA P3400PS
Laser-Drucker
400dpi Auflösung
73 Schrifttypen, PostScript
6 MB RAM, 20 MB Festplatte
2 x 250 Blatt Papierkassette
Rabenschwarzer Druck
12 Seiten/Minute
Interface:
RS232, Centronics, Apple Talk
Preis : DM 25.490,-

Siegfried Theimer GmbH
Rohlerstr.10, 6484 Birstein 6
Tel. 06054/8030 Fax. 06054/2619 Tlx. 4184314

FOTOPLOTTER

Wir machen Ihren Penplotter zum Fotoplotter!



Fotoplot-Zusatzgerät LP2002, geeignet zum Anschluß an jeden Flachbett-Plotter. Gleichbleibende Schärfe und Strichbreite durch geschwindigkeitsabhängige Steuerung der Lichtintensität. Hervorragend geeignet zur Herstellung von Leiterplatten-Filmen mit CAD-Systemen.

Fotoplot-Zusatz LP2002 .....2223,00 DM

Fordern Sie Informationsmaterial an!

Datentechnik Dr.Gert Müller Diezstraße 2A
D-5300 Bonn 1
Tel. 0228/217297

XYZ-STEUERUNG

Schrittmotoren positionieren unter *HPGL

Modulares System, ausbaufähig bis zu 3 Achsen, steuert Schrittmotoren bis zu 2A Arbeitsstrom - anschlußkompatibel zu ISERT-Anlagen. Der eingebaute Rechner (68000) besitzt eine Centronics-Schnittstelle und ermöglicht so z.B. CNC-Fräsen, -Bohren und Gravieren unter direkter Kontrolle von PC/AT CAD-Systemen wie AutoCAD®).

3 Achsen komplett (ohne Schrittmotoren) **..... 1980,00 DM**

Datentechnik Dr.Gert Müller Diezstraße 2A
D-5300 Bonn 1
Tel. 0228/217297

* HP-GL ist ein Warenzeichen der Hewlett-Packard GmbH

Industrie- u. Meßkarten für PC, XT, AT

– 8-Bit-Analog-Digital-Wandlerkarte	129.–
– 12-AD/DA-Karte mit 16/1 Kanal, 0–9 V	248.–
– 35-MHz-Analog-Digital-Wandlerkarte, 8 Bit	548.–
– Relaiskarte mit 8 TTL-I/O und 8 Relais bestückt	249.–
– Optokopplerkarte mit 16 Ein- und 8 Ausgängen	368.–
– Prototypenkarte mit 24 TTL-I/O	198.–
– 48-I/O-Karte mit 3x16-Bit-Timer und LED-Anzeige	248.–
– Schrittmotorenkarte für 2 Stepper	298.–
– EGA-/BAS-Wandlermodul, wandelt RGB in SW-Video	119.–
– Frequenzzählerkarte bis 1,3 GHz	298.–
– VD-8008, Videodigitalisierer, 1024 Pixel, 256 Graustufen	991.–
– BUS-EXTENDER-Karte für Servicearbeiten	98.–
– Slot-Erweiterung für PC/AT, 4 Steckplätze außerh.	256.–
– Netzwerkkarte EASY-Network mit LAN u. Software kompl.	290.–
– Modem-Karte, 300/1200 Baud, kompl. mit Software	339.–
– Akustikkoppler mit 300 Baud von Tandy	245.–
– Telefax-Karte kompl., Gr. 3 DBP, max. 9600 Baud	2900.–
– Drucker-Puffer-Karte mit 64 KB	264.–
– RS-232 ser. Schnittstelle, erweiterbar auf 2	89.–
– RS-422-Schnittstelle komplett	338.–
– IEEE-488-Schnittstelle komplett	780.–
– RS-232 auf RS-422-Wandlermodul, im Gehäuse	498.–
– 384-K-Multifunktionskarte, Uhr und sämtl. Schnittst.	189.–
– Multi-serielle Karte mit getrennten RS-232-Schnittst.	229.–
– Bar-Code-Pen mit Interface, viele Normen	791.–
– Handy-Scanner, Bildesegerät, 62 mm breit	819.–
– Maus, MS-Mouse-kompatibel, u. Treiber	99.–
– Genius-Maus mit Treiber und Paintsoftware auf CGA	130.–
– GAME-I/O-Karte für PC mit 2 Eingängen	79.–
– Joystick (analog) für PC	36.–
– Eprommer LOW-COST bis 27256 Textoolsock. auf Karte	298.–
– Eprommer-Karte bis 27512 u. Textoolsock. extern	398.–
– Eprommer mit 4 Textoolsockeln extern	599.–
– PAL-Programmer mit 1 Textoolsockel	679.–
– 2-MB-RAM-Karte für XT, 0 K best., mit Treibersoftw.	329.–
– 386-AT-TOWER-Gehäuse u. 200-W-Netzteil	899.–
– CCD-1000, Industriekamera mit 500 x 500 Pixel, ab 3 LUX	1799.–
– 9-Zoll-Monitor, passend zur CCD-Kamera	248.–

Wir sind Hersteller von PC-Karten. Andere Karten auf Anfr. Bitte kostenl. INFO anfordern. Händleranfragen erwünscht. Versand erf. per NN. Auch EXPORT bei Vorscheck mögl.

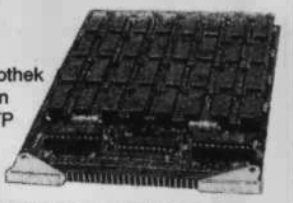
COMPUTER-RING Heinrich Kolter Electronic
Steinstraße 22 · D-5042 Erftstadt
Tel. 0 22 35/7 67 07 · Fax 0 22 35/7 20 48 · Tx 8 882 134

STÖHR HARDWARE SOFTWARE

ADD-MEM HP

1 Mbyte Speichererweiterung für
HP Laserjet Serie II
Canon LBP-8 II
2 Softfonts im Lieferumfang (jeweils 8, 10 Punkt)
Preis DM
850.-

Komplette Bibliothek von zusätzlichen Softfonts für DTP verfügbar



Deutsche Entwicklung und Fertigung

Stöhr Hardware Software
Wildbacher Mühle 3 * 5100 Aachen
Tel.: 0241/172371 Fax: 0241/172891

ELINK-03 Datenfernübertragung ohne Probleme

- 300 BAUD vollduplex
- 1200 BAUD halbduplex
- 1200/75 BAUD vollduplex
- 75/1200 BAUD vollduplex
- ELINK-03 wählt vollautomatisch und nimmt Anrufe entgegen (MAILBOX)
- SMARTMODEM kompatibel: arbeitet mit Standardsoftware wie OPEN-ACCESS, SYMPHONY, KERMIT, CROSSTALK usw.
- Eingebaute Intelligenz erledigt Datentransfer mit jedem Computer oder Terminal über V-24-Schnittstelle, natürlich mit automatischer Baudratenerkennung
- Einfache Programmierung für Spezialaufgaben durch den Anwender
- Betrieb an Nebenstellenanlagen möglich
- Postzulassung mit FTZ-Nummer
- günstige Mengenstaffeln
- komfortables MAILBOX-Programm ebenfalls erhältlich

MICOM Computer- und Informationssysteme Entwicklungs- und Vertriebsges. mbH
Wüstenhofer Str. 6 · 5600 Wuppertal 1 · Tel. 02 02/44 34 01

mc-AllProg Plus

Universelles Programmiergerät aus mc 6-8/1987
mit erweiterter Software

+ Programmiert jetzt auch GALs.
  PROMs von TI und Zeropower-RAMs
+ Vorbereitet für Single-Chip-Prozessoren von INTEL
+ Unterstützt das INTEL-Hex-Format
+ Inhaltsverzeichnisanzeige

* Programmiert (E)EPROMs bis 27512, 27513 und 27011, PROMs von Signetics/Valvo und PALs von MMI, NS, TI und AMD
* Anschluß über RS 232 (V.24) Schnittstelle
* Bediensoftware für IBM-PC/XT/AT (unter PC/MS-DOS), kompatible und Z80-Systeme (unter CP/M) verfügbar
* Bildschirmorientierter (E)PROM-Editor
* Bildschirmorientierter Text-Editor zum PAL-Entwurf
* PAL-Assembler + PAL-Simulator + PAL-Disassembler
* Fuselisten als JEDEC-Dateien speicher-/lesbar

# Bausatz	(inkl. Handbuch und Software)	997,– DM
# Fertigplatine	(getestet, Handb., Software) ..	1420,– DM

Versand per Nachnahme. Bei Bestellung unbedingt Diskettengröße, Diskettenformat und Betriebssystem angeben!

Holger Haase & Michael Menrad
Peripheriegeräte GbR
Postfach 2647 / 3300 Braunschweig
Telefon: (0531) 400310

mc-QUICKIE

mc-quickies sind aktuelle Produktanzeigen, mit denen Firmen ihre Produkte vorstellen. Verantwortlich für den Inhalt sind die Inserenten.

Der Eprommer für **IBM** und -kompatible **Apple** //e, //gs,][+, und **CPC 464/664/6128**

✱ Programmiert alle gängigen EPROM- und EEPROM-Typen (z. B. 2716, 27C16, 2732, 2732A, 27C32, 2758, 2764, 2764A, 27C64, 27128, 27128A, 27C128, 27256, 27C256, 2508, 2516, 2532, 2564, X2804A, X2816A, X2864A ...) ✱ Menügesteuerte Software auf Diskette/Kassette ✱ 32 KByte frei für EPROM-Daten (Brennen des 27256 ohne Nachladen) ✱ Kein Umschalten, Stecken oder Löten nötig ✱ Programmierspannungen werden im Gerät erzeugt ✱ Verbindung zum Rechner über Flachbandkabel ✱ Rote und grüne LED zur Betriebsart-Anzeige ✱ Komplett mit 28pol. Textool-Sockel ✱ IBM- und CPC-Version mit Interface-Karte (durchgeführter Expansionsport bei CPC-Version) ✱

Preise für IBM: Fertiggerät **DM 399.50** ✱ Bausatz **DM 349.–**
für Apple: Fertiggerät **DM 269.50** ✱ Bausatz **DM 219.–**
für CPC 464/664: Fertiggerät **DM 289.50** ✱ Bausatz **DM 239.–**
für CPC 6128: Fertiggerät **DM 319.50** ✱ Bausatz **DM 269.–**
CPC-Software auf 3"-Diskette statt Kassette: + DM 15.–

DOBBERTIN
INDUSTRIE-ELEKTRONIK **GmbH**
Brahmsstraße 9, 6835 Brühl, Tel. (0 62 02) 7 14 17

DIN-A3-Plotter PLX-88

Spitzentechnologie zum Sensationspreis

Made in Germany

Preis DM **1498,-**

- DIN-A3-Plotter
- 4 Farben m. autom. Stiftwechsel
- 0,025 mm Auflösung
- max. 200 mm/s Zeichengesch.
- HPGL-Kompatibel 28 Befehle
- Serienmäßig zwei Schnittst.
  Centronic u. RS 232 (V.24)
  ohne Aufpreis
- Ladbare Zeichensätze

CMC GmbH
Gesellschaft f. Electronic u. Microprozessorsysteme mbH,
Sebastianstraße 5, D-8480 Weiden, Telefon 0961/35053

Konvertierungsprobleme?

lösen Sie mit der Hard- und Software von

A.S.S.-WARE

Mainframe *1) <–> MS-DOS
CP/M *2) <–> MS-DOS
MS-DOS *3) <–> MS-DOS

Die Konvertierungssoft- und Hardware läuft auf XT- und AT-kompatiblen Rechnern unter allen MS-DOS-Versionen ab DOS 2.X und 3.X.

*1) Z. B. Basisdatenaustauschformat: IBM 3540, 3601, 3602, 3740, 3741, (= IBM/370) 3742, 3747, 3962, 5110, 5114, 5231, 5320, 5340, 5380, Honeywell Bull, SEL usw.
IBM /32, /34, /36, /38. 8" und 5.25"-Disketten.
Nixdorf 8870

*2) Fast alle CP/M-Formate von 3" bis 8"

*3) Fast alle MS-DOS-Formate von 160 KB bis 2 MB natürlich auch PS/2 1440 KB netto

Fordern Sie bitte ein Info an.

A.S.S.-WARE, Alfred Herrmann, Schimmelshahn,
5461 Roßbach/Wied, Tel. (0 26 38) 45 13

Zeichnungen

einfach mit dem Computer erstellen!
E-CAD-Software CG87 der neuen Generation erstellt aus Ihren Angaben: Schaltungs-/Platinenvorlagen, Verbindungsliste, Stückliste, Bestückungs-, Bohrplan mit höchstauflösender Grafikkarte bis 1200 × 873 Pixel in 16 Farben, lauffähig auf IBM PC-XT/AT.
Leistungsmerkmale: Mehrlagentechnik, SMD-fähig, Symbol- u. Makrobibliotheken, Gummibandtechnik, Logiktest, Fangroutinen, Auflösung bis 1/1000 Zoll. Entflechtungshilfen: optimierte Schaltungs-, Layout- und Autorouter.
Ausgaben: Plotter, Photoplot, Laserdrucker, Matrixdrucker, Diskette, CNC-Steuerungen.

Programme zur Layout- und Schaltbilderstellung gibt es bereits für **DM 968.–.**

Photoplotservice v. CAD FILE schnell u. preiswert.

Grafikkarten

Höchstauflösend mit bis zu 1280 × 1024 auf dem Bildschirm sichtbaren Punkten und bis zu 256 Farben pro Punkt. Bildspeicher bis 2048 × 2048 Punkte, für IBM- o. ECB-Bus-Systeme
inkl. Software **ab DM 990.–.**

Info anfordern bei:
Klaus Gorny · Computer Soft & Hardware
Bülowstraße 16b · 1000 Berlin 30 · Tel. (0 30) 2 16 91 29

LAYTRONIC

PCB-FULL-SERVICE

- *Film-plots für alle CAD- und Low-cost-CAD-Systeme im HP/GL- und Gerber-Format.*
- *Film-plots auch in 24 Stunden.*
- *Plot-Service mit Musterprints im Eildienst.*
- *Klein- und Mittelserien aus eigener Fertigung.*

LAYTRONIC GmbH · Industriestr. 7, 7434 Riederich
Telefon 07123/33635, 34223 · Telex 7245409, Fax 33282

NEU

miniMODUL-535

Das universelle Mikrocontroller-Board in Scheckkarten-Größe mit Siemens 80535 Controller

- CPU-Platine für Siemens 80535/80C535
- 8 x 8/10 Bit Analog-Eingänge
- 24 parallele I/O Leitungen, Adress-, Daten- und CS-Leitungen für kundenspezifische Erweiterungen verfügbar
- 16 Bit Watchdogtimer, 3 x 16 Bit Timer/Counter
- Akkupufferung für CMOS RAM, Power-down-Schaltung
- 64 k RAM + 64 k EPROM, wahlweise EEPROM
- Serielle Schnittstelle
- Einsetzbar als großer Chip (2 x 32-polig) in Ihrer Applikation
- Umfangreiche Software wie Prozeßbasic, Monitorprogramm, Makroassembler, C-Compiler, Pascal-Compiler und Simulatoren für MS-DOS-Rechner verfügbar

Ihre Vorteile:

- durch konsequenten Einsatz der SMD-Technik erhöhte Zuverlässigkeit und geringer Platzbedarf
- sehr günstige Preise
- weniger Aufwand bei Lagerhaltung und Einkauf
- kürzere Entwicklungszeiten
- schneller von der Idee zum Produkt

Preise: ab DM 198,– + MwSt. (DM 225,72 incl. MwSt)

PHYTEC
PHYTEC Meßtechnik · Vogelsbergstraße 74 · Postfach 705
6500 Mainz 42 · Tel. (0 61 31) 50 88 71 · Telex 4 187 192

Ecosoft Economy Software AG

Postfach 1905, Abt. A36, 7890 Waldshut, Tel. 077 51 - 79 20

Prüf-Software und Frei-Programme (fast) gratis

- Größtes Angebot Europas: Über 3'000 Disketten für IBM-PC/Kompatible, Macintosh, Amiga, Atari ST, C64/128, Apple II.
- Viele deutsche Programme, speziell für Firmen, Selbständige, Privatpersonen, Schulen.
- Software gratis, Sie bezahlen nur eine Vermittlungsgebühr von DM 14,40 oder weniger je Diskette.
- HOTLINE: Tel. Anwenderunterstützung (3 Techniker von 8-18 Uhr).

Verzeichnis gratis

Bitte Computermodell angeben.

HW ELEKTRONIK

HANDELS GMBH

2000 Hamburg 20 Eimsb. Chaussee 79
TEL. 040 / 439 68 48 u. 430 00 19
NN-VERSAND ab DM 30.– + Versandspesen
Zwischenverk. vorbeh. *Erf.-ort Hamburg

EIN KLEINER AUSZUG AUS UNSEREM REICHHALTIGEN ANGEBOT

TEAC - LAUFWERKE			DATA-T-SWITCH SUB-D		6303	23,50
FD 35 FN	228,00		701 m.Schnittst.-test	29,50	63 B 09	29,50
FD 35 GFN	298,00		1xIN/2xOUT	129.–	63 B 09E	29,50
FD 35 GFN 29	318,00		702 dto.1xI/4xO	169.–	63 B 21	13,50
FD 35 HFN	299,00		707 1xIN/2xOUT	48.–	63 B 40	14,50
FD 55 B-R	227,00		708 1xIN/4xOUT	59.–	63 B 45	29,50
FD 55 F-R	227,00		713 Centr 2xOUT	48.–	68000-8	16,30
FD 55 GF-R	275,00		714 dto. 4xOUT	69.–	68 HC 000-8	59.–
GT 35 Ver.-Set	69,00		X-SWITCH 2xIN/2xOUT		68000-10	22,50
Manual	je 15,00		803 Bu SUB-D 25p	89.–	68000-12	34,50
NEC - LAUFWERKE			806 Bu Centr.	99.–	68010-8	47,50
FD 1037 A	198,00		SMART-SWITCH SUB-D		6856A N8	84,50
FD 1054	288,00		605 1xIN/4xOUT	239.–	6256A N5	77,50
FD 1137	326,00		606 1xIN/8xOUT	329.–	68590	147,00
FD 1157 C	238,00		PRINTER-BUFFER 9V:		68591	62,50
D 5126	669,00		601 1xOUT 64K	224.–	68681	16,90
D 5126 H	899,00		602 2xOUT 64K	299.–	68881-12,5	335,00
D 5146 H	1299,00		603 1xOUT 256K	399.–	68901	27,90
Contr. AT	329,00		INTERF.-CONV. 8 C/9V		8052 AH 1.1	59,50
Contr. XT	189,00		607 parall./ser.	139.–	8087-5	195,00
Manual	je 15,00		608 ser./parall.	139.–	8087-8	319,00
3 M-DISKETTEN p.10 St			PR.-KABEL f. IBM	9.–	80286-10 LCC	199,00
3,5" SS	39,90		TEXTOOL-NULLKRAFT-S.		80287-8	479,00
5,5" DS	45,50		16p 19,90 20p	20,50	80287-10	575,00
3,5" DS/HD	105,00		24p 18,75 28p	18,95	80386-16	799,00
5,25" SS	19,90		28p schmal	49,50	80386-20	1399,00
5,25" DS	27,90		40p 37,50 64p	109.–	80387-16	1075,00
5,25" DS-96tpi	36,50		ATARI-FLOPPY-STE	3,95	80387-20	1899,00
5,25" HD-96tpi	49,50		ATARI-MONIT.-STE	2,95	8250 B	12,50
3,5"Reinig.-Set	15,90		SCART-STECKER	3,50	TMS 4500A15	36,50
5,25" dto.	13,50		HONDA-STECKER	5,95	µPD 70108-10	33,50
EXP.-PC-KARTEN			Stromvers. 3,5"	1,50	µPD 70116-10	34,50
f. "APPLE"	14,50		dto. 5,25"	2,50	V 40-8	89,00
f. "IBM XT"	39,50		KARTENSTE 20p	2,30	V 50-8	99,00
					V 25	

mc-QUICKIE

mc-quickies sind aktuelle Produktanzeigen, mit denen Firmen ihre Produkte vorstellen. Verantwortlich für den Inhalt sind die Inserenten.

Pocketmodem 1200 **498,–**
Nur 10 × 6 × 2,4 cm klein, für Portabels usw.

Modemkarte PC 1200 **349,–**
Kurze PC-Karte, V21/22, 300/1200 Bd vollduplex, RS 232 eingebaut (COM 1–4), Lautsprecher.
KS 1200, extern, sonst wie PC 1200 ..... **398,–**
KS 2400, 2400/1200 Bd vollduplex **598,–**
Multimodem, V21/22/23 ..... **598.–**
Alle Modems Hayes-komp. und **ohne FTZ-Nr. Anschluß an das Postnetz nicht erlaubt.**

RAMboard 2 MB **349.–**
EMS-/Intel-komp., ohne RAM, mit Software
AT-RAMboard mit 16-Bit Slot, sonst w. o. **449,–**

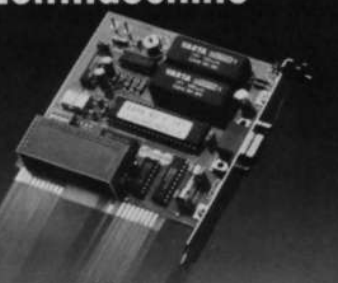
AD/DA-Wandler **269,–**
Auflösung 12 Bit, 16 AD-, 1 DA-Kanal

VIA 8255 **179,–**
I/O-Karte, 48 Leitungen, 16 LED, Timer/Counter

ARClink-Netzwerkkarte 790,–
Z. B. für Novell, Info über weitere ARClink-Produkte anfordern

bsb
Hohenzollernring 74
5000 Köln 1
Telefon **02 21-13 14 41**
Datentechnik GmbH

Die Zeitmaschine



Das Funkuhrensystem DXTA von AUERSWALD hat alle Vorteile:

- akkugepuffert
- belegt keine Schnittstelle
- für alle PC/XT/AT
- Software für alle Anwendungsbereiche incl. Quelltext
- sofort betriebsbereit, mit Antenne und Diskette
- **nur DM 375,– (incl. MWST)**

Versand per NN, zuzügl. Versandkosten

Auerswald GmbH & Co. KG
Vor den Grashöfen 1
3302 Cremlingen
Tel. 0 53 06/20 21
Tx. 952453, Fax 71 79

AZTEC **C – COMPILER** **MANX**

- Native-Compiler für MS-DOS, CP/M80, CP/M86, Apple, Amiga
- Cross-Compiler für 80186/286, Z80/8085, 6502 und 68000
- Compiler, Assembler, Linker, Hex-Converter, HLL-Debugger
- alle Tools zur Eprom-Erzeugung sind enthalten

Schema II **Schaltplan – CAD**

- superschneller Bildaufbau mit Zoom und Scrolling
- DESIGN-RULE-CHECK, Stücklisten- und Verbindungslisten
- Bauteile - Bibliotheken für Elektronik und E - Technik

PROTEL-PCB **Platinen – Layout**

SPITZENLEISTUNG ZUM SUPERPREIS !!!

- manueller Router mit 1/1000 Zoll Auflösung
- unterstützt Multilayer, SMDs, Gerberplots und Excellon
- automatische Netzlistenkontrolle mit Soll/Ist - Vergleich
- interaktiver Autorouter als Erweiterungspaket

Demodisketten für SCHEMA und PROTEL je DM 50,--

INGENIEURBÜRO DIPL.ING. MANFRED SUCHY
Gottlieb-Daimler-Str. 12 8037 Olching
Telefon 08142/12360 und 08142/28028

GWK GESELLSCHAFT FÜR TECHNISCHE ELEKTRONIK mbH

NEU
68020 – 68881
für
Selbstbaucomputer
c't 68000 GWK

Weltweit leistungsfähigstes Systemkonzept für modulare Systeme im Einfach-Europa-Format

Kostenloses Informationsmaterial anfordern!

Postfach 1360, D-5120 Herzogenrath,
Telefon (0 24 06) 60 35, Telex 8 32 109 gwk d

Das kostenlose Christiani Kursprogramm

Lernen Sie heute, was morgen unentbehrlich ist.

Fordern Sie gleich das kostenlose Kursprogramm an, das Sie über unsere Lehrgänge informiert – u. a. auch über:

- ☐ Elektronik-Labor
- ☐ IC-Labor
- ☐ Digital-Labor
- ☐ Oszilloskop-Labor
- ☐ Fernseh-Labor
- ☐ Elektronisches Messen
- ☐ Mikroprozessortechnik
- ☐ Peripherie-Bausteine
- ☐ Kompakt-Kurs BASIC
- ☐ BASIC + Mikrocomputerpraxis
- ☐ Amateurfunk
- ☐ Automatisierung
- ☐ Optoelektronik-Labor
- ☐ El. Steuerungstechnik
- ☐ Videotechnik
- ☐ Kompakt-Kurs Elektronik
- ☐ CNC-Technik
- ☐ C64 CNC-Simulator
- ☐ Bautechnik
- ☐ EDV-Grundlagen

Gewünschtes ankreuzen; Anzeige ausschneiden und einsenden an

Dr.-Ing. P. Christiani · Technisches Lehrinstitut
Postf. 35 69160 · Tel. 0 75 31/58 01-0 · 7750 Konstanz

SOFTWARE & HARDWARE
für Apple II & Mac & MS-DOS-Computer

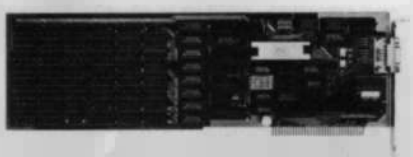
z. B. für APPLE II und Kompatible
Wir liefern die RAMworks-Karte (AE) für den Apple IIe mit max. 16 MB (z. B. erweitert Appleworks) z. B. RAMWORKS III 64 K ..... DM 469.–
Anpassung für Appleworks 1.2 auf dem II+/e. Original oder mit externer Tastatur. Anpassung in deutsch
für SATURN 128 K und IBS AP33 1 MB! ..... DM 130.–
AD 16 Ch. 12 Bit, schnell! (Applied Eng.) ..... DM 799.–
PKASO/U-Printer-Karte (IS) ..... DM 429.–
Timemaster II H. O., **die** Uhrenkarte! (AE) ..... DM 299.–
ELF kompl. Statistik-Software (TWG) ..... DM 495.–
Prime-Plotter-Grafik-Software (Primesoft) ..... DM 629.–
Z-RAM Ultra 3, 256 K, Z80 CP/M + Uhr, max. 1 MB ..... DM 798.–
Z-RAM Ultra 2, 256 K, mit Uhr, max. 1 MB ..... DM 698.–

z. B. für IBM und Kompatible
MatchPoint-PC plus UNIFORM-PC (Lesen/Schreiben von Apple Disks im IBM PC/XT/AT/100 % komp. für DOS-ProDOS-CP/M) ..... DM 649.–
MatchMaker (Anschluß von Mac 800 K Drive an IBM PC/XT/AT/100 % komp. zum Datentransfer Mac-MSDOS) ..... DM 498.–
CompatiCard (Anschluß von 3.5", 5.25", 8" Drives an IBM PC/XT/AT/100 % komp. z. B. 3.5" 720 K/1.44 MB) ..... DM 629.–
XENO-PC (Cisin) (Lesen/Schreiben von CP/M- u. MS-DOS-Formaten im IBM PC/XT/AT/100 % komp.) ..... DM 258.–
ELF PC kompl. Statistik Software (TWG) ..... DM 500.–

z. B. für alle Systeme
Printerchanger 3 parall. Drucker auf 1 Micro inkl. Kabel/Netzteil (Keyzone) ..... DM 340.–
Printersharer 3 Micros auf 1 parall. Drucker inkl. Kabel/Netzteil (Keyzone) ..... DM 340.–

Wir sind Import-Spezialisten und bieten Ihnen eine große Auswahl an Software und Hardware bedeutender Hersteller aus den USA und England. Informationen gegen DM 3.– in Briefmarken.

WEISS COMPUTER **Dipl.-Psych. Karl-Heinz Weiß**
Am Wiesenhof 17, 2940 Wilhelmshaven, Tel. 0 44 21/8 31 79



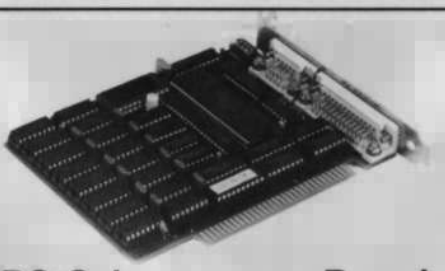
hochauflösende Farb-Grafik

für IBM/XT/AT (u. Komp.) mit SC-64

- NEC-7220, 8 MHz Highspeed-Version
- 1024 × 1024 Pixel, 8 Farben, 3 Mbit RAM
- 64 MHz oder 40 MHz Videotakt
- weitgehend kompatibel zu Artist 1
- Softwaretreiber in Pascal, C, usw.
- SC-64 Grafik-Board

Mehr als 500mal bei bekannten CAD-Systemen u. anderen grafischen Anwendungen im Einsatz., Made by SC.

SC SPECIAL COMPUTER GMBH
LW. Hauptstr. 5 D-7630 Lahr
Tel. 07821/4990 Telex 754848.



PC-Subprozessor-Board

für IBM (kompatible) PC, XT, AT

- CPU HD64180, Takt 4,6 MHz
- 512 KByte RAM, max. 4 KByte EPROM
- Zwei serielle Schnittstellen, vollduplex
- Input 5 Bit parallel (Maus, Lesestift usw.)
- Dipschalter (3 Bit, von Software lesbar)

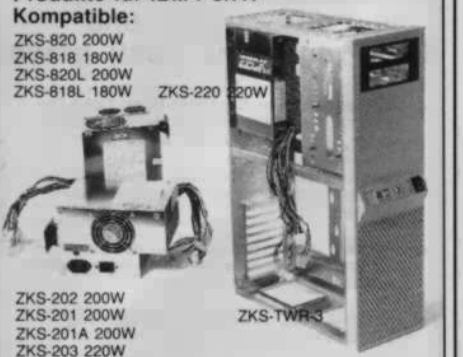
Viele Anwendungen, z. B. intellig. Druckerpuffer, Barcode, Datenerfassung usw.

SC SPECIAL COMPUTER GMBH
LW. Hauptstraße 5 D-7630 Lahr
Telefon 0 78 21/49 90 Telex 7 54 848

ZECK **FOR 386 TOWER CASE AND SWITCHING POWER SUPPLY**

Produkte für IBM-PC/AT Kompatible:

ZKS-820 200W
ZKS-818 180W
ZKS-820L 200W
ZKS-818L 180W



ZKS-202 200W
ZKS-201 200W
ZKS-201A 200W
ZKS-203 220W
U.P.S. ZPS-500 * Distributor Wanted

ZECKS ENTERPRISE CO., LTD.
No. 38,35 LANE SEC 6 CHUNG SHAN N RD. TAIPEI
TAIWAN R.O.C. FAX: 886-2-8311744 TEL: 886-2-8342525
TLX: 29266 ZECK

mc-QUICKIE

mc-quickies sind aktuelle Produktanzeigen, mit denen Firmen ihre Produkte vorstellen. Verantwortlich für den Inhalt sind die Inserenten.

BASIC-52 Rechnersystem mit AMS-BUS

● Folientastatur mit FIP-Anzeige 2 × 20 ● Netzwerkfähig über RS485 (Master/Slave) ● RS232 Host und RS232 Printer-Port ● SYSLIB Assemblerbibliothek ● TOOL-52 für PC mit Editor, Terminal, Renumber und Programmdownload
● Kundenspezifische Konfigurationen

LIPPERT AUTOMATIONSTECHNIK GMBH
D-6800 Mannheim 1
Krappmühlstraße 34
Telefon 0621/402012

Hannover Messe Industrie, Halle 13, Stand A 24/28

VIDEODIGITALISIERER
f. IBM PC, XT/AT u. Kompatible

VD 1951: Schneller, hochauflösender 1-Bit-Echtzeitdigitalisierer. Scanfrequenz 12 MHz. Digitalisierung in 1/50 s. Hochwertiger 70-MHz-Eingangsverstärker. 3 Eingangskennlinien, 2 Graumuster (grob u. fein) über Tastatur wählbar. Kontrast, Helligkeit, Graustufen (max. 5) stufenlos regelbar. Auflösung: Hor. 770 Pixel, Ver. 311 Zeilen. 512 KBit RAM on Board. Option: Externe Triggerung. Karte belegt 1 Slot. Umfangreiche Softwareunterstützung in Turbo-Pascal (teilw. Sources) für IBM-CGA, Hercules- und EGA-Grafikkarte.
Best.Nr: VD 1951 (Digitalisierer u. Software) DM 2100,–

VD 2000/8: 8-Bit-Echtzeitdigitalisierer. 256 Graustufen. Input Look-Up-Table (LUT), Ausgangsfarbgenerator (CLUT). 256 KByte RAM on Board. Alle anderen Daten wie VD 1951. Umfangreiche Softwareunterstützung in Turbo-Pascal (teilw. Sources) für Hercules- und EGA-Grafikkarte.
Best.Nr.: VD 2000/8 (Digitalisierer u. Software) DM 4100,–
Info gratis, Demo-Disk DM 20,– (V-Scheck od. Schein).

DIPL.-ING. OTTMAR WEGNER
Adalbertstraße 25, 2300 Kiel 1, Telefon 04 31/33 38 77

3", 3,5", 5,25", 8"
SS, DS, SD, DD, HD
PC/XT/AT
360/720 KB – 1,2/1,44 MB

Max. 4 Drives, gem. Betrieb, Umschaltung hardware- oder softwaremäßig (Option). Anschlüsse: 34pol. Standard und 50pol. 8". Kurze Karte. Lesen/Schreiben der Formate 360/720/1200/1440 KB (AT/S2-Format) mit mitgelieferter Software. Fremdformate mit Spezialsoftware.

Der Supercontroller für die Datenkonvertierung DM 495.–

MITSUBISHI-Laufwerke
MF355C, 3,5", 1/2 MB DM 280.–
M2896, 8", 1,6 MB DM 1200.–
MF504, 5,25"-Multifunktionslaufwerk, 40/80 Track, 0,5/1/1,6 MB DM 270.–

WEGE ELEKTRONIK
Hans-Detlef Wege
Grubenstraße 4, 4130 Moers 3,
Telefon bis 19 Uhr 0 28 41/7 20 38

MCU MCU MCU

AT-Kompatibles System 1799.–
CPU 80286, 8/12 MHz.
1 MB RAM bestückt 512 KB, Gehäuse, 200 W Power, Tastatur, Hercules-kompatible Grafikkarte, parallele Schnittstelle, Disk-Controller, 1x1,2 MB Laufwerk.
mit 20 MB Festplatte 2530.–
Tower-Gehäuse mit Power 700.–
XT-Kompatibles System .... ab 1060.–
EGA-Karte 249.–
Monitore ..... ab 269.–
386-System ..... ab 4999.–
386-Board 3399.–
16 MHz, 0 MB bestückt, 1x32 Bit Slot
Bei kompletten Systemen liefern wir Ihnen zusätzlich 10 Programmdisketten (Textverarbeitung, Datenbank, Kalkulations-Terminalprogramm, Spiele u.v.m.) mit. Auf Wunsch stellen wir Ihnen gerne jede gewünschte Konfiguration zusammen. Bitte fordern Sie unseren Computer- und Zubehör Katalog sofort an.
Händleranfragen erwünscht (mit Händlernachweis)
Tel. 089/3514636

Münchner Computer Unternehmen
8039 Puchheim, Bäumlstraße 22

Sie haben Probleme?
Wir haben die Lösung
Unterbrechungsfreie Stromversorgungsanlagen

z. Beispiel: **USV-MUK**
Leistung: 0,5 kVA
Überbrückungszeit: 7 min
„Slim line"-Ausführung
Nur 2100,– DM

z. Beispiel: **USV-MUD**
Leistung: 1 kVA
Überbrückungszeit: 15 min
Dauerbetrieb mit Sinusausgang
Nur 3850,– DM

Seit mehr als 40 Jahren sichere Stromversorgung von
JOVY-ATLAS SPECHT
STROMRICHTER GMBH
D-2960 Leer · Groninger Straße 29-37 · Postfach 15 80
Tel. (04 91) 60 02-0 · Tx 2 77 71 jovyle-d · Fax (0491) 6002-10

moVe
Mohwinkel und Veiser GmbH

UNIX SYSTEM V
für AT + 386 AT

AT+T-Lizenz ab ..... DM 592.–

DOSMerge 286 (DOS-Programme wie z. B. LOTUS unter UNIX) ab ..... DM 444.–

Komplettes Entwicklungspaket mit Runtime + Software-Development ab ..... DM 1200.–

Bitte fordern Sie unsere detaillierten Unterlagen, auch zu anderen PC-kompatiblen Produkten, an.

5090 Leverkusen 1 · Berliner Str. 73
Postf. 25 01 66 · Tel. (02 14) 9 37 81-9 50 60

PANASONIC KX-P 1081

STIFTUNG WARENTEST - TEST 2/88:
Im Test: 18 Nadeldrucker.
Testurteil: 7 gut*, 7 zufriedenstellend, 4 mangelhaft.
*Davon Panasonic KX-P 1081
gut

Hervorragende NLQ-Schrift in vielen Varianten, Schnellschrift mit 120 Z/Sek., umschaltbarer ASCII und IBM Zeichensatz, Centronics IF, eingebauter Traktor, Walze für Endlospapier und robuste zuverlässige Bauweise machen den KX-P 1081 zum preiswerten Drucker für viele Anwendungen.

KX-P 1081 - Unser Preis: . . nur 495,– DM

THEO WEBER ELEKTRONIK, 8700 WÜRZBURG
Eisenbahnstraße 22, Tel. 09 31-70 14 41

PANASONIC KX-P 1083

STIFTUNG WARENTEST - TEST 2/88:
Im Test: 18 Nadeldrucker.
Testurteil: 7 gut*, 7 zufriedenstellend, 4 mangelhaft.
*Davon Panasonic KX-1083
gut

Mit max. 240 Z/Sek., hervorragender NLQ Schrift, ASCII und IBM Zeichensatz, Centronics IF, eingebautem Traktor und Walze für Einzelblatt ein besonders schneller und zuverlässiger Universaldrucker. Natürlich auch Epson kompatibel.

KX-P 1083 - Unser Preis: . . nur 795,– DM
Vollautomatischer Einzug dazu: . **340,– DM**

THEO WEBER ELEKTRONIK, 8700 WÜRZBURG
Eisenbahnstraße 22, Tel. 09 31-70 14 41

AKTUELLE SONDERANGEBOTE:

Seagate Festplatte ST 225 - 21 MB .... 435,– DM
Seagate Festplatte ST 251 - 43 MB .... 895,– DM
Seagate Festplatte ST 4096 - 80 MB .. 1890,– DM
WD Controller 1002A-WX1 mit Kabel ... 145,– DM
Omti Controller 5520A mit Kabel ..... 145,– DM
Archive Fastape FT20 Backupsystem .. 1295,– DM

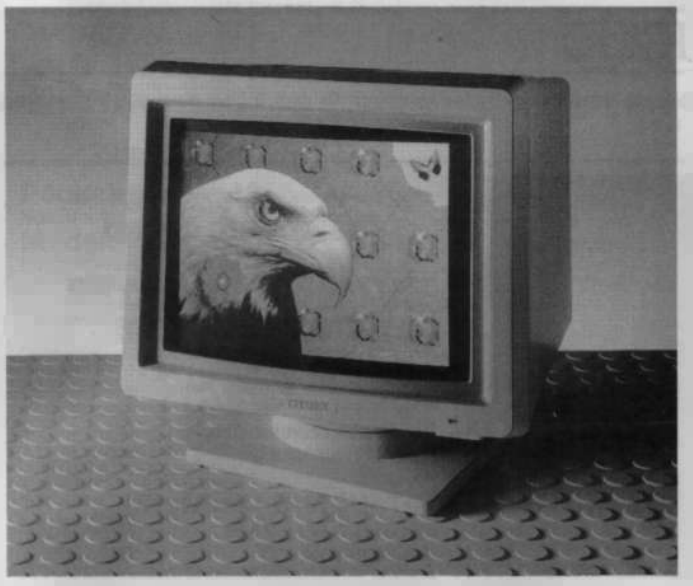
SEKONIC Plotter SPL 410 - A3, 6 Farb. 1980,– DM
SEKONIC Plotter SPL 450 - A3, 8 Farb. 2250,– DM
SEKONIC Plotter SPL 430 - A3, 6 Farb. 2980,– DM

NEC Multisync II - neuestes Modell ... 1495,– DM
Getronics MC6710, 15,5-35 KHz. 14" .. 1250,– DM
NEC Multisync II mit Genoa Super Ega 1995,– DM
ATI EgaWonder Enhanced ..... 450,– DM
Genoa Super Ega HiRes + GEM Graph 695,– DM

VICTOR 14" TTL Monitor bernstenfarben 350,– DM
Getronics FM 1400 - 14" Monitor, weiß, . 360,– DM

Wir liefern Ihnen natürlich alle PANASONIC Drucker für jeden Leistungsbereich. Außerdem bieten wir Ihnen ein vielseitiges Lieferprogramm an Computern, Matrix- und Laserdruckern von OKI und NEC, Plottern und allem gängigen Zubehör. Gerne senden wir Ihnen auch unseren umfangreichen Katalog. Bitte anfordern!

THEO WEBER ELEKTRONIK, 8700 WÜRZBURG
Eisenbahnstraße 22, Tel. 09 31-70 14 41



Alternativ zum IBM Monitor 8514 bietet sich der preiswertere CCM 104 von Citizen an

Farbmonitor von Citizen

Steckerkompatibel zum IBM-Farbmonitor 8514 und damit direkt an die IBM PS/2-Computer Modell 50, 60 und 80 anschließbar ist der Farbmonitor CCM 104 von Citizen. Der 14-Zoll-Monitor kostet mit 3000 DM deutlich weniger als das Original. Die höchste Auflösung erreicht der CCM 104 mit 1024 × 768 Punkten. Als Bildröhre setzt der Monitor das Trinitron-Modell von Sony ein. Citizen gibt eine zweijährige Garantie auf den CCM 104, der eine Lebensdauer von etwa 20000 Betriebsstunden haben soll. Für den Betrieb des Monitors mit XT/AT-Computern unter EGA, CGA oder VGA wird es von Citizen eine Grafikkarte für etwa 1500 DM geben.

Synelec
Lindwurmstr. 95
8000 München 15
☎ 089/51790

Programm sucht den optimalen Baustein

Eine Datenbank, in der alle zur Zeit verfügbaren PLD-Bausteine (Programmable Logic Devices) mit den notwendigen Betriebsparametern gespeichert sind, kann eine sehr gute Ergänzung zum CAE-System LOG/iC sein, dessen PLD-Compiler den Anwender beim Schaltungsentwurf unterstützt. Als Entwickler spart man sich nun einerseits das Blättern in Datenbüchern, andererseits muß man sich beim Schaltungsdesign zunächst nicht auf einen bestimmten Chip festlegen: Die Datenbank macht passende Vorschläge, basierend auf den Optimierungsergebnissen des Compilers. Gibt man zusätzlich Eigenschaften ein, wie z.B. die maximale Stromaufnahme oder Geschwindigkeit, wird das Auswahlnetz immer engmaschiger, und als Resultat bleibt meist ein einziger Baustein übrig. Auch das Finden kompatibler Typen mit ähnlichen Eigenschaften

ELCO ELECTRONIC COMPONENTS GmbH

KOMPAKT-AT 10 MHZ

- CPU 80286-10 (80287 Option)
- umschaltb. 6/10 MHz, 0 Wait State
- aufrüstbar bis 16 MB (1MB on Board)
- 512 KB bestückt
- Uhr/Kalender batteriegepuffert
- Mono/Graphic/Printer-Karte (Herc.)
- Parallele Schnittstelle
- DIN Tastatur mit sep. Cursor/Z.-Block
- 1.2 MB Laufwerk
- Norton SI 11.5/Landmark 13
- stabiles Einschubgehäuse

ohne Monitor **DM 1.798,-**

KOMPAKT-AT 13 MHz

- Konfiguration wie oben ● CPU 80286-12
- SI 15.3, Landmark 17,3

2198,-

TURBO-XT 4.77/10 MHz

- voll PC/XT kompatibel
- CPU 8088-1/Sockel f. 8087
- 8 XT-Slots/256 KB RAM
- 1x360 KB FDD
- 150 Watt Schaltnetzteil
- Color Graphic Karte o. HGC/P
- Parallele Druckerschnittstelle
- Serielle Schnittstelle RS-232/Game Port
- Echtzeituhr/Kalender batteriegepuffert
- DIN Tastatur mit sep. Cursor/Z.-Block
- Stabiles Einschubgehäuse/ LED/Schlüssel

ohne Monitor **DM 998,-**

Andere Rechnerkonfigurationen auf Anfrage!

- Typenraddrucker Silver Reed/20 Z/sek. ..... **DM 467,—**
- Epson LX-800/incl. Interface/Handbuch ... **DM 599,—**
- Mitsubishi 1471 ..... **DM 1498,—**
- 14" TTL Mon., entspiegelt, m. Standbasis ... **DM 278,—**

ELCO *Electronic Components GmbH*
D-6460 Gelnhausen-2 · Am Spielacker 18
☎ 06051/66088
Telex: 4184524 hzbm d · Fax: 06051/69205

mc-MINIMARKT

Debugger f. 8048-Familie 285 DM
½ Europa-Mikrocomputer 171 DM
EPROM-Emulator 855 DM
D. Krause, Amsterdamer Str. 31, 7030 Böblingen,
☎ 0 70 31/27 42 35

IBM-Branchensoftware für **Radio/TV** und **Fliesenhandel, z. B. mit Kalkulation,** Serienbrief usw. **ind. Anpassung.** ☎ 0 21 02/5 14 47

Endlich ist es da – das Public Domain und Shareware-Handbuch, auf das Sie schon lange gewartet haben.
PD-Handbuch (90 S.) DM 5.– oder PD-Handbuch mit 2 Begleitdisk.
– Disk A: Program Reviews
– Disk B: Intro Disk DM 14.80
Lieferung per NN (+ DM 6.50) oder Vorkasse (zuzügl. DM 2.– Porto) auf Postscheckkonto Mchn., Kto.-Nr. 3403 13-807, Euroscheck oder Briefmarken.
REDYSOFT Software GmbH, Postf. 12 61, 8150 Holzkirchen G

Haftetiketten, 40 Größen. Info: Böhne, Försterweg 4, 3354 Dassel G

Unglaublich, aber wahr! Über 1000 Disketten mit PC-Freiprogrammen ab 5 DM lieferbar. Der neue Katalog ist schon da, kostenlos, wie's immer war. Telefon rund um die Uhr 0 52 61/8 89 01, wagen Sie's nur. Günther, Braker Mitte 28C, 4920 Lemgo G

SUCHE HARDWARE

Suche günst. PC-AT, neu od. gebraucht. Angebote an R. Fröhlich, Drususstr. 14, 6472 Altenstadt, ☎ 0 60 47/43 17

Habe noch 2 St. TERM1, suche GRIP2 mit GRIPS.
EBA, Aulweg 129, 6300 Gießen, ☎ 06 41/7 02 45 89

IQS-401 FFT-Analyzer f. Apple II gesucht. Zuschr. unt. Nr. MC 101

Suche MICE II für 6809.
☎ 0 79 51/2 60 63

Suche Floppy EPSON PF-10 oder TF-20 oder TF-15.
☎ 0 79 61/45 45

BIETE AN HARDWARE

mc68000, CPU68010, 8 MHz, 512 KByte, voll funktionsfähig, 500 DM. ☎ 0 89/5 02 52 93

NDR-Klein-Computer: Verkaufe 2 TEAC-LW 5¼", FD-55-F, TAST 2 (Preh), Zenith-Monitor, Bernstein, Geh. 2. ☎ 09 31/1 29 50

Transputerboard macht Ihren PC XT/AT zum Supercomputer, DM 2700.–. ☎ 0 69/5 48 44 71

MC68010, 1,5 MByte, 2 LW, Term-Uhren-EPROM Z80-Karte. OS-9/CP/M68K, Gedit, Utility, alles orig., wegen Hobbyaufgabe. Alles in Orig.-Gehäuse, 3500.– VS. ☎ 02 21/56 10 22, ab 18 Uhr

Verk. **HPLASERJET Serie II,** ¾ Jahr alt, Pr. VB. ☎ 0 24/8 48 88

RGB-PAL WANDLER, Platine, geprüft, 169.–. Fertigger. mit Netzt. 298.–. ☎ 02 21/73 76 09, ab 17 Uhr

CP/M-System, prof. aufgebaut, Industrie-Netzteil, Profi-Tastatur ACS, Grafikkarte, 68000CO-Platine, EPROM-Floppy, 5 ¼-Floppy 800K, E-Prommer, Software-Fachbücher, ☎ 09 51/7 11 55

Gewerbl. Anzeigen sind mit G gekennzeichnet.

MIDI für den Apple II! Neues, wesentlich erweitertes Interface und 4-Spur-Sequencer. Info geg. 1.30 Porto. J. Lange, Maßmannstr. 4/3, 8000 München 2

Verkaufe MC68000 im Gehäuse, mit 68010, 1,5-MByte-RAM, Sprach-/Uhrenkarte, Terminatorkarte, 2× Floppies 1 MByte, Monitor u. Tastatur, inkl. Software, CPM68K, Modula, Gedit, alles Original mit Handbücher, 12 000 DM, ☎ 0 23 23/2 39 87, werktags ab 19 Uhr

85-MB-HD+Contr. für MC68000, 2×FD, 1-MByte-BASF, Epr. und Sprach-/Uhrkarte, auch einzeln. ☎ 0 89/3 54 25 68

5¼"-Floppies: Shugart SA410 (80 T, SS), BASF6106 (60 T, SS), VHB je 120 DM.
☎ 0 30/3 41 21 08, ab 18 Uhr

EPROM-Simulator für 2716 bis 27512 (o. Geh.), **DM 248.–.**
Info bei: **HüWiCo,** Am Siepen 17, 4630 Bochum, ☎ 02 34/36 12 06 G

Temperaturmeß. m. Apple II, Hard- u. Software, 24.– DM, Info 1 DM. G. Gassner, vor der Höhe 11, 3320 Salzgitter 51

LOGIK-ANALYSATOR, Hewlett-Packard hp1610A, 32-Kanal, DM 2850.– VB. ☎ 0 62 98/32 28

12-MHz-IBM-AT, inkl. AutoCAD 2.6 Logikanalysator, nach Funkschau, preiswert zu verkaufen.
☎ 0 52 41/4 78 34

EPSON HX-20 mit int. 32 KByte-RAM, Drucker, Koffer, Literatur und Zubehör zu verk.
☎ 05 61/2 22 46

Siemens 6.610, 3 LW (je 1 MByte), Drucker Siemens 9003 u. Konsole, BS Amboss + C/PM, VB 4500 DM, dto. 2 LW (je 256 KByte), Drucker MT140, VB 3500 DM,
☎ 0 22 24/8 01 00, ab 19 Uhr

Eprommer // Teac Teac

mit ext. Textoolsockel im Gehäuse., Editor, Programmierspannung:: Vpp 25V; 21V und 12.5V Programmiert 2716 bis 27512 auch C Typen Lieferumfang: Steckkarte ext. Epromsockel, Software Beschreibung zum Preis von nur **248 DM**

Frei Programme

für IBM oder Kompatible Computer in deutsch und englisch weit über 1200 Disketten. Liste der Freiprogramme KOSTENLOS.

TEAC 55BR 360 KB 269DM 35FN-23 720KB 5,25"Rahmen 339DM
TEAC 55GFR 1,2MB 289DM 35HFN 1,44MB 299DM 5/88
TEAC 55FR 720 KB 289DM 35FN 720KB 279DM Hanbuchje 15DM
Handy Scanner 698DM HandyReader Texterkennung 898DM

UEDING electronics
Holtewiese 2 Tel. 02373 63159
5750 Menden 1 DFÜ 02373 66877 8n1
Inh. Gregor Ueding Gesamtpreisliste mit weiteren preisgünstigen Angeboten anfordern

SCHUKAT

Für Industrie und Fachhandel: Krischerstraße 27 · 4019 Monheim / Rhld. Tel.: 02173/50005 · Telex: 8515732 sele d

CO-PROZESSOREN INTEL

8087	5 MHz
8087-2	8 MHz
8087-1	10 MHz
80287- 6	6 MHz
80287- 8	8 MHz
80287-10	10 MHz
80387-16	16 MHz
80387-20	20 MHz

PREIS-GÜNSTIG

BACHER GMBH
Sendlinger Str. 64 · 8000 München 2
Tel. 0 89/26 50 94 · Telex 5 214 624

FORTH 83 Compiler

Interaktive Programmentwicklung für MS-DOS, MS-OS/2 oder 80386 mit Tools und gleicher Anwendungsschnittstelle.

Leistungsfähiger ROM-Anwendungsgenerator für folgende Zielprozessoren:
8031/51/515, Z80/HD64180, 8086/88/186, 8085, 8096/196, 6303, 6502, 1802, Z8, 68000, 6809 sowie 32 Bit 80x86, 680x0 und TMS34010.

Die Übersetzer laufen auf IBM-PC kompatiblen Rechner und ermöglichen über die serielle Schnittstelle ein interaktives Arbeiten auf der Zielhardware.

Der ROM/RAM Emulator SRS63 wird von der Software direkt unterstützt.

FORTH-SYSTEME
ANGELIKA FLESCH
Postfach 1103
D-7814 Breisach am Rhein
Telefon (0 76 67) 5 51
Telefax 07667 555

Alexander Janson

dBase für Profis

Ausgefeilte Programmier-Tricks für Insider mit dBase II, III und IIIplus.
331 S., 89 Abb.,
geb. DM 68,–
ISBN 3-7723-8871-X

Dieses Buch ist randvoll mit wertvollen Hinweisen und Ratschlägen, die der Leser sofort in die Praxis umsetzen kann. Es ist klar und übersichtlich in Themenbereiche gegliedert und beginnt da, wo die übliche dBase-Literatur aufhört.

Das Franzis-Fachbuch
Ein Qualitätsversprechen
Franzis-Verlag GmbH, Karlstraße 37–41, 8000 München 2

mc 5-88-22

mc-MINIMARKT

BIETE AN HARDWARE

NDR-Computer: Umbau- und Erweiterungsplatinen, alles getestet und funktionsfähig, zu niedrigen Preisen. ☎ 0 22 62/9 34 84 G

*** **AT + 386 Kompatible** ***
Baugruppen u. Fertiggeräte:
POWER-AT1:
16 MHz OWS, Speed 21.3
POWER-AT2:
13 MHz OWS, Speed 17.3
TURBO-AT1:
10 MHz OWS, Speed 13.0
TURBO-386:
20 MHz OWS, **Speed 26.3**
Ethernet-Netzwerke: 10 MBit/s
Unterlagen anf.: G. M. E-Versand, Postf. 10 01 01, 4048 Grevenbroich, ☎ 0 21 81/4 86 72 G

Habe noch 2 St. TERM1, suche GRIP2 mit GRIPS.
EBA, 6300 Giessen, Aulweg 129, ☎ 06 41/7 02 45 89

24 Nadeldrucker-NEC Pinwriter, 40 Tage alt, Orig.-verpackt, NP 1200 DM für 950 DM.
☎ 0 52 54/1 30 95, ab 17 Uhr

MegaBit-C-MOS-RAMs, 120 ns, 32 St. je 63.–; 256 KByte×1 C-MOS-RAMs, 70 ns, 40 St. je 21.50; Mengenrabatt.
☎ 0 30/4 04 55 04

OSBORNE 5 Turbo 640-KByte-RAM, 1×360-KByte-Laufwerk, MS-DOS 3.21, AT-Tastatur, Hercules-Kompat., Grafikk. 2×parallel, 1×seriell, 1×Gamep., Bernsteinmonitor, GW-BASIC, VB 1800.–.
☎ 02 28/66 45 64, nach 18 Uhr

MC68000 2.5-MByte-RAM, Bustermuhr, SASI-Interf., EPROM-Karte, Floppy-Contr., CPIM, BIOS 2.0, SW.
☎ 05 11/41 00-2 37, 0 57 24/70 46

Logikanalysator, 16 Kanäle/10 MHz, (Graf-El.), komfortabel, komplett mit Monitor, Tastatur usw. DM 1000.–.
Dieter Kriesell, Badgasse 6, 8531 Lonnerstadt, ☎ 0 91 93/10 43

MATRIX-DRUCKER
C.ITOH 1550, Druckbr. 136 Z/Z, Matrix 9×7, 120 Z/s, Grafik-Drukker, VB 800.–. ☎ **0 69/51 86 95**

● **Kamera** für Ossi und Monitor
● **Laborwagen**
Traumhafte Preise z. B.
● D. Multimeter ab DM 108,--
3 Stck. ab DM 98,--
● D. Multimeter TRUE RMS ab DM 450,--
● F. Generator ab DM 412,--
● P. Generator - Testbildgenerator -
● Elektron. Zähler ab DM 399,--
● Netzgeräte - jede Preislage -
● Meßkabel - Tastköpfe - R, L, C Dekaden
● Adapter - Stecker - Buchsen -
● Video-Audio-Kabel u. v. m.
Prospekt kostenlos
Händleranfragen erwünscht
HAMEG sofort ab Lager **HAMEG**
Bachmeier electronic · Danziger Str. 4 b
2804 Lilienthal ☎ 0 42 98 - 49 80
HAMEG sofort ab Lager **HAMEG**

Genie III S echte 8 MHz, 256-KByte-RAM Z-80H Luxus-Knürr-Gehäuse, integr. P39-Monitor, 512×512 Pixel, 2×80 Track, inkl. Meßrechner-Zusatzkarte, 8 MHz, Z-80H, 64-KByte-RAM, 3×8253, 3×8255, Bauteile einzeln mehr Wert! NUR DM 1400.–.
☎ 02 28/34 61 51

TELEX-ADAPTOR mit ZZF-Nummer für Existenzgründer und alte Hasen mit V24/V28 vernetzbar mit einer oder mehr elektr. Schreibmaschinen od. PCs. **ELEKTRONISCHER FERNSCHREIBER** (mtl. DM 96 Postgebühren) zum SUPERPREIS von **DM 3500.–** inkl. MwSt. TECHNO-CHEM HGMBH, Frankfurt,
☎ 0 69/74 98 98, Tx 4 170 532 G

COMPAQ PORTABLE III, AT-kompatibel, 1.1-MByte-RAM, Cop. 80287, VB DM 4345.–.
☎ 0 40/5 23 62 64, ab 18 Uhr, oder 0 40/5 25 93 64

Apple II+, 64 KByte, PC-Geh., ex. Tast., 2×5,25", Z80, 80Z, 68008, par., PAL, Monitor, DOS, PruDOS, 68008-Softw., div. Lit., VB 1750 DM. ☎ 0 68 31/4 97 39

*** CP/M ECB SYSTEM ***
Z80H-256-KByte-Single-Board ab 120.–
7220-Grafik 1k×1k×4 ab 120.–
1.5-MByte-RAM-Floppy ab 120.–
Info gratis von
Ing.-Büro Klaus Rehwald, Lindenstr. 25, 6290 Weilburg 4

MC-CP/M Plus, Witex-WordStar-Terminal, ADI DM-14 Monitor, IBM-PC-Geh., 2 TEAC FD-55-Laufwerke, EPROM-Programmierer, RAM-Platine, Fujitsu DPMG-9 Drucker, viel Software, einschl. Manuals., kompl. DM 2500.–.
☎ 0 24 21/50 15 51, ab 18 Uhr

mc68010-II, 2,5 MByte, SIO, EPROMMER, Z80, 20-MByte-Harddisk, CP/M, 05/9, viel Softw., 2998 DM. ☎ 0 25 01/28 78

mc-CP/M-Computer, Mon. bernst., super Key AN96FI, 2×8" Shugart, Eprommer, Tape, viel Software, **kompl. nur 999.– DM.**
☎ 0 40/38 68 46

Epson LX 800, neuwertig, VB 550.– DM. Chiffre MC 102

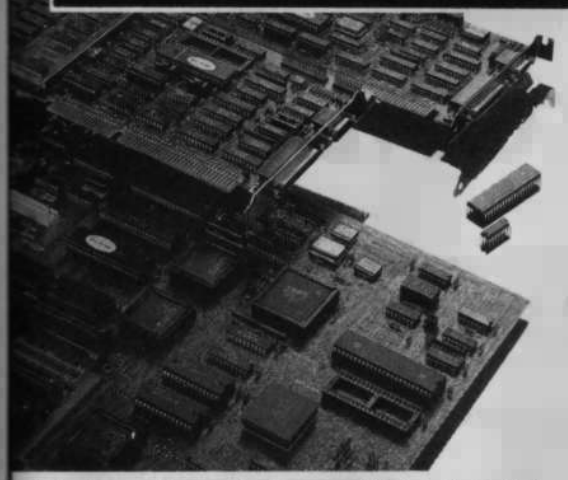
NDR68008, kpl. mit 2-Floppies-Geh., Mon., gr. Tastatur usw., VHB 2000 DM, auch einzeln.
☎ 09 11/60 15 82

NDR-Computer CP/M-Vollausbau, 2 Floppies, Turbo-Pascal, dBase 2 ZEAT, SER, Monitor, VB 1800.– DM.
☎ 0 83 67/4 72, ab 17 Uhr

Logic-Analyzer (Gould) K105-D, NP 40 000.–, 1 Jahr alt, für 25 000.–. ☎ 0 30/8 81 26 24

Einige gebrauchte Matrixdrucker Binder 8510A à 350.–. E-Techn.-Büro, ☎ 0 66 52/5 05-0 G

JUST FROM A SMALL CHIP



OEMS WELCOEM

Contact us today for more details!!

LEGOL BIOS/DOS

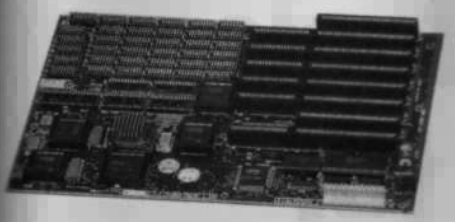
Tower system available! also.

We look for ecxellent quality from a small chip to the finished product, we currently produce 3,000 PC's and 20,000 interface cards each month. Our eight years of professional experience ensure quality products, all of which are backed up with a one year warrantly. Out engineers are always at your service.

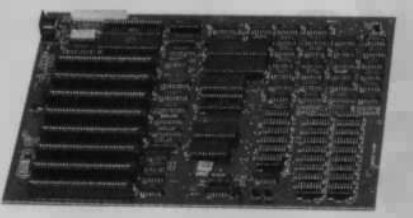
OUR PRODUCT LINE:
*88, 286, 386 Mainboards. *PC/XT/AT Interface cards
Covering: Ram Expandsion, I/O, new Generation Display, Modem, Eprom Writer, LAN Card Etc....

IBM, PC, XT, AT are registered trademarks of International Business Machines Corp.

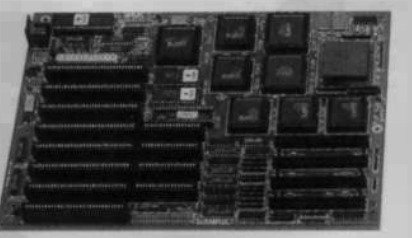
Err-Free ENTERPRISE CO., LTD.
P.O.Box: 91-414, Taipei, Taiwan, R.O.C.
Cable: ERRFREE TAIPEI, Telex: 19028 ERRFREE
Tel: (02) 7762285 (5 Lines) Fax: 886-2-7009027



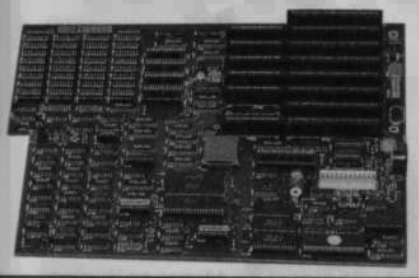
- CPU 80286-10, 6/12 MHz, 0/1 wait
- 1 MB RAM, Up to 4MB on board



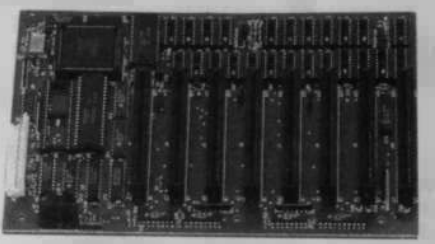
- CPU 8088-2, 4.77/10 MHz
- Speed control with sfotware & hardware switchable



- CPU 80386-16, 20 MHz, 0/1 wait



- CPU 80286-10, 6/12 MHz



- CPU 8088-1, 4.77/7.15/9.54 MHz
- Baby size 16 x 22cm

Cards Available:

SUPER EGA	AT MULTI I/O
MGP	FDC
CGA	FDC AT/XT
CGP	AT 2S/2P
XT MULTI I/O	

KEC KAN-EAN COMPUTER CORP.
2Fl, No. 10, Lane 30, Yen Gean St.
Taipei, Taiwan, R.O.C.
Fax: 886-2-7772351 Tel: (02) 7 77 23 50

Der **Franzis-Verlag** hat seit Jahrzehnten qualitativ hochwertige Elektronik-Bücher in seinem Programm. Wir wollen unseren Erfolg auch im Bereich der Computer-Literatur ausbauen.

Deshalb suchen wir kompetente

BUCHAUTOREN

mit guten Erfahrungen auf folgenden Gebieten:

- **Betriebssysteme**
- **Programmiersprachen**
- **Anwenderprogramme**
- **Hardware**

besonders für den Bereich der PC- und PS/2-Systeme.

Sie sollten mit uns zusammen ein attraktives Computer-Buchprogramm mit hohem Qualitätsniveau und hoher Aktualität aufbauen.

Wenn Sie bereits Erfahrung als Autor haben und auch Spaß daran haben, Ihr Wissen an andere in schriftlicher Form weiterzugeben, sollten Sie nicht zögern, mit uns Kontakt aufzunehmen.

Die Voraussetzung bei unserem gemeinsamen Vorhaben ist natürlich, daß Sie sich auf einem der obengenannten Gebiete fundiert auskennen.

Nehmen Sie Kontakt mit uns auf:

M. Heysinger, Chef-Lektor

Franzis-Verlag GmbH, Karlstraße 37, 8000 München 2, Telefon 0 89/51 17-2 42

MS-DOS für Fortgeschrittene

Das große MS-DOS-Profi-Arbeitsbuch

MS-DOS intern – für den Profi in allen Versionen aufbereitet. Mit PC-DOS 3.3.
Von D. Smode.
2. Aufl., 448 S.,
71 Abb., Tab. und Listings,
geb. DM 68,–
ISBN 3-7723-8682-2

Im Buch wurden alle Versionen bis einschließlich PC-DOS 3.3 berücksichtigt. Das Haupt-Augenmerk liegt nicht auf der Mensch/MS-DOS-Schnittstelle, sondern im Bereich Anwenderprogramm/MS-DOS.

Das Franzis-Fachbuch

Ein Qualitätsversprechen

Franzis-Verlag GmbH, Karlstraße 37–41, 8000 München 2

mc 5-88-20

Jetzt wieder lieferbar!

Basic für Aufsteiger

Der sichere Weg zum fortgeschrittenen Basic-Programm.
Von R. Busch.
3. Aufl., 285 S.,
74 Abb., 11 Tab.,
geb. DM 58,–
ISBN 3-7723-7283-X

Die meisten Basic-Einführungen hören nach einer ersten Kontaktnahme mit wenigen Befehlen auf. Wer weitermachen will, der hat das richtige Buch in der Hand. Es vermittelt nicht nur eine Menge an Wissen, es ist auch unterhaltsam zu lesen.

Das Franzis-Fachbuch

Ein Qualitätsversprechen

Franzis-Verlag GmbH, Karlstraße 37–41, 8000 München 2

mc 5-88-24

INSERENTENVERZEICHNIS

A.S.S-Ware 145
Abor 32, 33
AD Computertechnik 154
Adcomp 125
Auerswald 146

Bacher 176
Baron 151
Bauer & Wetzel 155
Behrendt 156
Bockstaller 143
bsb Datentechnik 146

CE-TEC 69
Cesys 152
Chicony 160
Christiani 146
Chung Yu 178
Computer Discount 178
Computerring Eckernförde 152
Computerring Kolter 144
Computer 2000 ..... 14, 15, 58, 59
Conex 32, 33

DaCom 153
Dalvo 157
Datex 121
De Uneo 170
Digitec 35
Disys 178
Dobbertin 145
DSM 8, 9
DVS 177

ECO Institut 23
EcoSoft 145
Elco 148
Elektronikladen 24, 25
Elitegroup 161
Elsa 155
Emis 145
Ensontech 163
Epson 2
Err Free 179

Forth 176
Franzis 167, 176, 180
Fujitsu 7

Gorny 145
Graf 21
GWK 146

Haase & Menrad 144
Heise-Verlag 177
HW-Elektronik 145

Indutronic 3
Intec 143
Interconsult 177
Interplanetary 170
Intra 162
IST 13
iSystem 34

Jovy-Atlas Specht 147
Jow Dian 159

Kacer 174
Kan Ean 179
Koga 169
Krischer 163
Kwem 17
Kyocera 167

Laytronic 145
Lippert 147

Mathes 164
McGraw Hill 171
MCI 92, 93, 183
Memory Electronics 142
Meyer, E. W. 32, 33
Micom 144
Microcomp 172
Micromint 171
Microsoft 28, 29
Microtech 18
MoVe 147
MPI 152
Müller, Dr. Gert 144
Münchner Computer Unternehmen .. 147

Omega-Link 143

Palm Springs 168
Pandasoft 175
Phytec 145
Print Technik 143

Ranfft 143
Rheintec 156
RIM 175
Röntgen 143
Rose 143
RWL 150

S + M 142
Sanwel 158
Schukat 175
SC Special Computer 146
SE Spezial Elektronik 11
Shamrock 135
Siemens 54, 55
Sparkasse 37
Stöhr 144
Suchy 146

Tetzlaff 143
Theimer 142, 144
Toshiba 45
Trans Computer 168
TSS Schmitz 143

Ueding 175

Vobis 184
Vogel-Verlag 149

Weber 147
Wege 147
Wegner 147
Weiss 146
Western Digital 27
Wilke 142

Zecks 146
Zenith 31

Die Mikrocomputer-Zeitschrift

Verlag: Franzis Verlag GmbH, Karlstraße 37–41, 8000 München 2
Postfach 37 01 20, 8000 München 37
Telefon (0 89) 51 17-1
Telex 5 22 301
Telefax (0 89) 51 17-3 79

Geschäftsführer:
Peter G. E. Mayer, Michael-Alexander Mayer

Verlagsleitung: Peter Habersetzer, Eugen Wintersberger

Anzeigenleitung: Dietger Kötter

Anzeigenverkaufsleitung: Hans-Joachim Hecht (-3 86)

Disposition: Ingrid Daschner (-2 97)

Anzeigenvertretung Inland:

Norddeutschland: Lita Lange, impulse medien service GmbH, Holtenklinker Str. 9, 2050 Hamburg 80, Tel. 0 40/7 24 20 37/38
Hessen: Günter Junne, Victor-Achard-Str. 30, 6380 Bad Homburg v. d. H., Tel. 0 61 72/3 33 94
Bayern: Elfie Rusch, Münchner Verlagsvertretung, Sperberstr. 23, 8000 München 82, Tel. 0 89/4 30 73 32
Nordrhein-Westfalen: Bandelow + Partner GmbH & Co. KG, Herr Werner Bandelow, Dellestr. 36, 4000 Düsseldorf 12, Tel. 02 11/20 14 64
Baden Württemberg: Ulrich G. Felger, Honoldweg 27, 7000 Stuttgart 1, Tel. 07 11/63 27 18
Berlin: Rainer W. Stengel, Bischofsgrüner Weg 91, 1000 Berlin 46, Tel. 0 30/7 74 45 16

Anzeigenvertretung Ausland:

Belgien: ECI/United Media International S.A., Avenue de la folle chanson, 2 bte 7, 1050 Bruxelles, Tel. 02/6 47 31 90, Telex 63 950 eci um
Großbritannien: Martin Geerke, Friary Hall, Flat 3, Firary Road, South Ascot Berks. SL5 9HD, Tel.: 9 90/2 86 49 oder 2 76/6 28 28, Telex 8 58 328
Japan: International Media Rep. Ltd., 2–29, Toranomon 1-chome, Minato-ku, Tokyo 105, Tel. 5 02-06 56, Telex 22 761
USA: DemoNet Inc., 7310 Adams Street, Paramount, CA 90723, Tel. (2 13) 4 08-19 68
Frankreich: Agence Gustav Elm, 41, Avenue Montaigne, 75008 Paris, Tel. 01-47 23 32 67, Telex 2 90 260
Italien: Rancati advertising, Milano San Felice Torre 5, 20090 Segrate, Tel. 02-7 53 14 45, Telex 3 11 250 PP MI I
Schweiz: Exportwerbung AG, Kirchgasse 50, 8024 Zürich, Tel. 01-47 46 90, Telex 8 12 765

Anzeigen nach Preisliste Nr. 8, gültig ab 1. 10. 1987

Vertriebsleitung: Peter Habersetzer

Abonnement: Christa Fischer

Handelsverkauf: Dietlind Stauder

Erscheinungsweise: Die mc erscheint monatlich, jeweils montags am Monatsanfang bzw. am Ende des Vormonats; im 8. Jahrgang

Bezugspreise Inland: Einzelheft 7.– DM, Jahresabonnement 70.– DM; Vierteljahresabonnement 18.50 DM. Studenten, Auszubildende und Rentner erhalten das Jahresabonnement gegen Nachweis günstiger. Der Versand ist im Abonnementspreis eingeschlossen. In den Preisen ist die gesetzliche Mehrwertsteuer in Höhe von 7 % enthalten.

Verlagsvertretungen Ausland:

Belgien: Office International des Périodiques (O.I.P.), Avenue Marnix 30, B-1050 Brüssel – Einzelheft 182.– bfr, Jahresabonnement 1963.– bfr.
Dänemark: Harck + Gjellerups Booksellers Ltd., Fiolstraede 31–33, DK-1171 Kopenhagen K. – Jahresabonnement 316.– dkr.
Frankreich: Librairie Parisienne de la Radio, 43, rue de Dunkerque, F-75010 Paris.
Luxemburg: Messageries Paul Kraus, 5, rue de Hollerich, L-Luxembourg.
Niederlande: De Muiderkring BV, Electronics House, Postbus 313, 1380-AH Weesp – Einzelheft 8.85 hfl, Jahresabonnement 89.50 hfl.
Österreich: Erb-Verlag Ges.m.b.H. & Co. KG, Buch- und Zeitschriften-Vertrieb, Amerlingstr. 1, A-1061 Wien. – Einzelheft 60.– öS, Jahresabonnement 620.– öS.
Schweiz: Verlag Thali AG, CH-6285 Hitzkirch/Luzern – Einzelheft 7.– sfr, Jahresabonnement 67.20 sfr (je nach Kurs).

VORSCHAU

Mini-Module

mc hat immer schon die Einplatinencomputer gern gehabt. Mit diesen „Bausteinen" kann der kreative Ingenieur vieles messen, regeln oder steuern. Es gibt fast kein Gebiet, in dem die Mikrocontroller auf den Einplatinencomputern nicht eingesetzt werden könnten. mc bringt einen Überblick über die Entwicklung auf dem Gebiet der Mikrocontroller – unter Betonung der 805X(X)-Modelle – und präsentiert eine Mini-Platine (Scheckkarte) als Steckmodul. Mit preiswertester Cross-Software!

Spitzenmodell

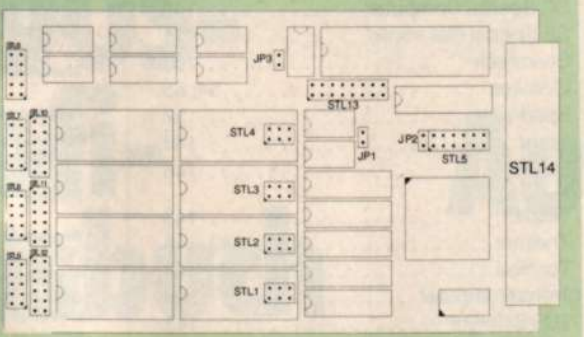
Die Konkurrenz betont, daß sich der PS/2-Standard noch längst nicht durchgesetzt hätte, IBM setzt seine neuen Modelle derweil besser und besser ab und die Firmen in Taiwan stehen alle Gewehr bei Fuß, um den Zeitpunkt nicht zu verpassen, ab dem „Mikrokanal-Klones" gefahrlos in den Markt gebracht werden können. Modell 80 von IBM.

VGAs im Test

Der Kontakt zwischen Benutzer und Computer läuft zum großen Teil über den Bildschirm und das Auge. Grund genug, immer wieder die neuesten Video-Karten ins kritische Visier zu nehmen. Diesmal zeigen zwei VGAs, was sie können. Dank der stürmischen Entwicklung der Technik wird die Computerwelt immer farbiger.

Transputer, Teil 3

Auch diejenigen, für die es nicht in Frage kommt, eine Transputerplatine selbst zu installieren, werden ihre Freude an diesem dritten Teil der Serie haben. Es geht um die Software für die Transputer. mc zeigt Schritt für Schritt, wie man die Transputer in Assembler programmiert. Befehl für Befehl ist erläutert.



Laserdrucker

Wir wollten eine Marktübersicht über Drucker ins Heft 6 bringen. Die Zahl der Antworten auf unsere Fragebögen war so gewaltig, daß wir die Marktübersicht „Drucker" teilen mußten. In Heft 6 werden wir 70 Laserdrucker mit Kurzbeschreibung, ca.-Preis und Bild vorstellen. Damit Sie ein vollständiges Bild des Angebotes bekommen. Die Laserdrucker erobern mehr und mehr Marktanteil ...

Aktuelle Sammeldisketten zu mc

MS-DOS-Sammeldisk 12
Diskette zum „PC-Sonderheft" (u. a. CLK.ASM, BAUM.ASM, TERM.BAS, DRUCKER.ASH, Hercules-Grafik, PC-Datentransfer, EGA-Grafik, Gameport, Disk-Speedy, Disk-Reparatur, Musik nebenbei). Preis: 25 DM.

MS-DOS-Sammeldisk 13
Diskette zum mc-Sonderheft „Turbo-Pascal im Einsatz" mit Pascal Quellcode.
Preis: 25 DM.

MS-DOS-Sammeldisk 14
Cross-Assembler und EPROM, Move-Utility für MOS-DOS und Zeitanzeige in Fenstertechnik aus mc 5/87; Wahrheitstafel für Logik-PROMs und schneller Draw-Line-Algorithmus aus mc 6/87; Hardcopy von CGA-Farbgrafiken und Flächen-Füll-Algorithmus aus mc 7/87; vier Bildschirmseiten in Turbo-Pascal, MS-DOS mit Komfort und Extended Memory (> 1 MByte) beim IBM-AT aus mc 8/87. Preis: 25 DM.

MS-DOS-Sammeldisk 15
256K-EGA mit variabler Bildgröße, Hercules-Grafiken drucken, Physik und Gehirn (KI) und Backup ganz bequem aus mc 9/87, Primzahlen-Sucher, pseudozufällige Binärfolgen, Utility für die Hercules-Karte und Leiterbahnen entflechten aus mc 10/87.
Preis: 25 DM.

MS-DOS-Sammeldisk 16
Differentieller Dateivergleich, dynamische Bildausgabe, PC minimiert Schaltfunktionen und Turbo-Residenz (aus mc 11/87). MS-DOS-Filemanager und Treiber-Implementation in MS-DOS (aus mc 12/87). Außerdem: Jahresinhalt mc 1987 mit Editor. Preis: 25 DM.

MS-DOS-Sammeldisk 19
Turbo-Pascal ohne Absturz, Linien in C (aus mc 1/88); GALs programmieren (aus mc 1–2/1988); Festplatte parken (aus mc 1/88). Preis: 25 DM.

Für den ATARI-ST gibt es vier Disketten:

ST-Sammeldisk 1
EPROM-Programmer und Taschenrechner aus mc 9/87 (TOS-Version vom 27. 11. 86 oder später nötig). Preis: 25 DM.

ST-Sammeldisk 2
Grafikdemo und Minimon aus mc 5/87.
Preis: 25 DM.

ST-Sammeldisk 3
3-D-Darstellung von Funktionen aus mc 7/87, Assembler-Routinen in GFA-Basic und Setzen des Uhrenbausteins aus mc 9/87.
Preis: 25 DM.

ST-Sammeldisk 4
Physik und Gehirn aus mc 9/87, Splitspeed und Modula-2-Co-Routinen aus mc 11/87, diverse Public-Domain-Programme (Achtung: zweiseitig formatiert). Preis: 25 DM.

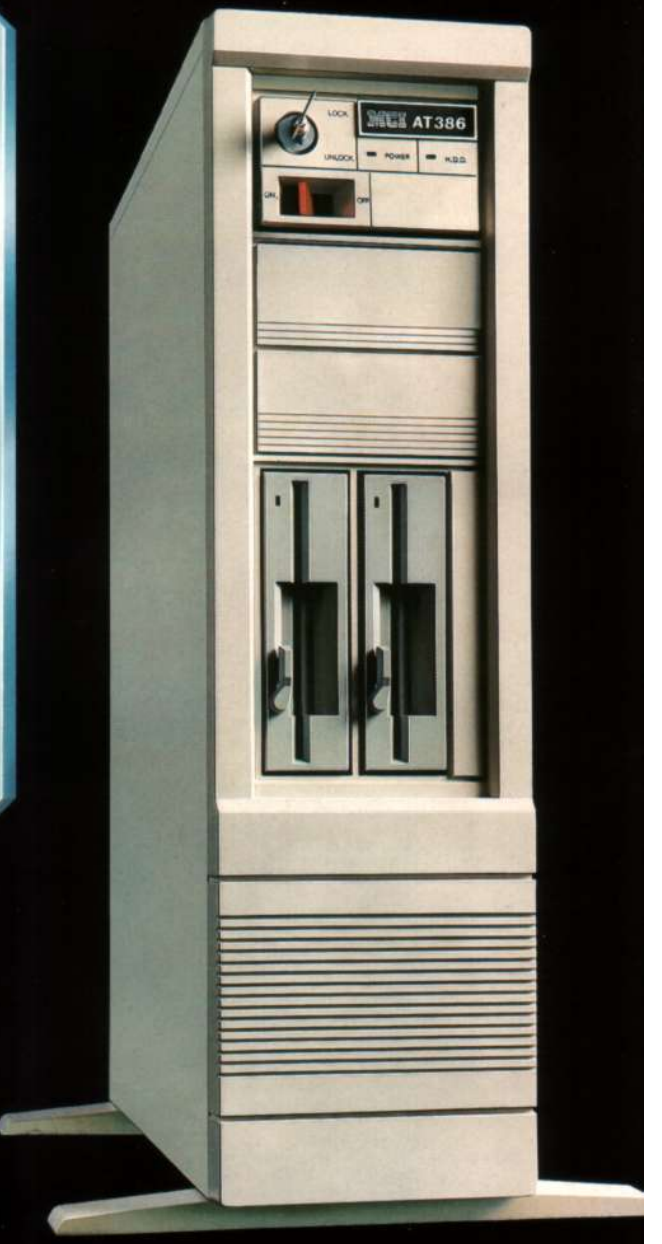
MS-DOS-Sa.-Disk 20: Know-how-Computer (mc 4/88)

Bestellung ☎ 0 89/59 54 68

Ausgabe 6/88 erscheint am 24. Mai 1988

MCI

PERSONAL COMPUTER



Bitte neueste Prospekte anfordern!

Auf alle Geräte 12 Monate Garantie. Preise gültig ab 1.12.87. Lieferbedingungen auf Anfrage. MCI MICRO COMPUTER INSTRUMENTS GMBH eingetragen AG Bergisch Gladbach HRB 2575 - Herstellung und Vertrieb von Microcomputern.

MCI

Bensberger Straße 252 · 5060 Bergisch Gladbach 2
Tel. (02202) 1080
Fax: (02202) 31009 · Telex: 8873518

Gekonnt preiswert!
Bei VOBIS, Deutschlands umsatzgrößtem Microcomputer-Spezialisten!
TEXAS INSTRUMENTS TI 66
150 Funktionen max. 512 Programmschritte oder 64 Speicher
99.-
Auch bei PORST
TEXAS INSTRUMENTS
TI 95 »PROCALC«
200 Funktionen 7200 Programmschritte
444.-
299.-
BASIC-programmierbar 8 K Speicher
TI 74 »BASICALC«
PC 324 Drucker f. TI 74 + 95
259.-
KP 548
Farbmonitor
648.-
Commodore
C 64 +
+ BTX-Modul
BTX DECODER MODUL II
Komplett nur
999.-
Commodore
AMIGA 2000 incl. KP 548 Stereo-Farbmonitor
2595.-
SIDECAR
PC-Erweiterung für A 1000
999.-
AMIGA 1000 komplett mit SIDECAR
nur noch
1999.-
ZENITH Z 148
IBM-kompatibel
1499.-
Mit 20 MB-Platte
1975.-
HIGHSCREEN
KP 548
Stereofarbmonitor
RGB RGBI Video Scart
Universell!
Nur
648.-
Auch bei PORST
Commodore
PC 10-III
Nachfolger des PC-10-II incl. SW-Monitor
1895.-
Mit EGA-Karte + EGA Monitor
2795.-
HIGHSCREEN
LCD 286
Der tragbare AT
Incl. 20 MB-Platte
3498.-
Auch bei PORST
Lingo
TOWER-AT
2 MB Ram, 80 MB-Festplatte incl. EGA-Karte + EGA-Monitor
7995.-
HIGHSCREEN
EP 720 EGA
EGA-Farbmonitor, 720 x 350 Punkte
798.-
CITIZEN IDP 560
40 Zeichen/Zeile
2-farb Drucker
Für Commodore VC nur
95.-
Auch bei PORST
SEIKOSHA GP 100 AS
Matrixdrucker mit serieller Schnittstelle
199.-
SEIKOSHA GP 700 A
8-farb Drucker
299.-
HIGHSCREEN
TV-Tuner für Farbvideomonitore
199.-
Auch bei PORST
CENTRONICS
GLP-Drucker
IBM-kompatibel, NLQ
Nur
365.-
Commodore
MPS 1200
Matrixdrucker NLQ für C 64, 128, 128 D
495.-
Auch bei PORST
Commodore
MPS 1000
Drucker für Commodore VC's und PC's
648.-
MULTILIFE
Qualitätsdisketten 100 % Error-free
10 Stck. ab
9.95
SEIKOSHA
SL 80
24-Nadel-schönschreibdrucker
SL 80 AI (IBM)
SL 80 VC
748.-
NEC P6
1095.-
Auch bei PORST
NEC P6 C(olor)
1345.-
Auch bei PORST
MULTIMATIC Einzelblatteinzug dazu
399.-
Der breite Drucker zum schmalen Preis!
NEC CP7 DINA3
24 Nadel-Schönschrift, DIN A3 breit, schwarz/weiß + farbig
1695.-
Artikel mit diesem Zeichen
Auch bei PORST
erhalten Sie auch in über 30 Filialen der PORST AG in BERLIN, HAMBURG, LÜBECK, HANNOVER, GÖTTINGEN, KREFELD, MÜNSTER, BOCHUM, WUPPERTAL, REMSCHEID, FRANKFURT, WIESBADEN, KRIFTEL, NEUNKIRCHEN, PIRMASENS, MANNHEIM, STUTTGART, HEILBRONN, KARLSRUHE, LÖRRACH, SENDEN, INGOLSTADT, ROSENHEIM, REGENSBURG, NÜRNBERG, SCHWABACH, BAMBERG, COBURG, SCHWEINFURT, AUGSBURG, MEMMINGEN.
WICHTIG! Herstellerbedingte Lieferzeiten! Auf Grund erhöhter Nachfrage ist nicht immer alles lieferbar!
VOBIS MICROCOMPUTER
kompetent und preiswert
VOBIS MICROCOMPUTER · POSTFACH 1778 · 5100 AACHEN · TEL. 0241/50 00 81 · TELEX 832 389 vobis d